SYSTEM ARCHITECTURES LABORATORY
DEPT. OF ELECTRICAL AND COMPUTER ENGINEERING
GEORGE MASON UNIVERSITY
Fairfax, VA22030

SCALABLE ADAPTIVE ARCHITECTURES FOR MARITIME OPERATIONS CENTER COMMAND AND CONTROL

ONR Award No: N00014-08-1-0319

January 16, 2008 – April 30, 2011

FINAL TECHNICAL REPORT

By

Lee W. Wagenhals
Abbas K. Zaidi
Alexander H. Levis

Submitted to:
Office of Naval Research
Attn: **Dr. Jeffrey G. Morrison**
Code 341
825 N. Randolph St., Suite 1425
Arlington, VA 22203-1995

Submitted by:
Alexander H. Levis
Principal Investigator
alevis@gmu.edu
Phone: 703 993 1619
Fax: 703 993 1601

Scalable Adaptive Architectures for Maritime Operations Center Command and Control

ONR Award No: N00014-08-1-0319

January 16, 2008 – April 30, 2011

FINAL TECHNICAL REPORT

ABSTRACT

This is the final technical report of the contract entitled "Scalable Adaptive Architectures for Maritime Operations Center Command and Control" that was conducted by George Mason University between January 16, 2008 and April 30, 2011. The overall objective of the effort was to study and analyze the new concepts of the Navy Maritime Operations Center (MOC) focusing on the scalability aspects of the MOC concept and develop tools and techniques that could support the design and evaluation of augmentation strategies that will allow MOCs to change or adapt to new missions and new levels of command and control. Specifically, the effort focused on developing an understanding of the issues involved in determining the fundamental principles and factors involved in changing the scale of a MOC through augmentation including information needed to design and analyze how to change a MOC's organization structures and underlying architecture as its mission changes.

We accomplished both objectives through a set of six tasks. A suite of tools was upgraded and modified that can support the modeling and analysis of MOC scalability. Several workflows were developed that support the process of using the tools to enable the design of MOC augmentation strategies. Techniques for considering the impact of cyber contested environments and space limitations on command vessels were developed. The tools were integrated into a test bed called the C2 Wind Tunnel. Scenarios were created and evaluated using the tools to demonstrate the tools and workflows.

A task was added later in the project to investigate the possibility of using earlier work on the validation and verification of rule bases in addressing the dynamically changing rules of engagement of a maritime force as it crosses different geographical areas. GMU developed an approach that uses a formal model checking methodology to detect redundant and inconsistent cases in a set of rules selected from across different Maritime Laws. The approach can provide considerable help to a Maritime Lawyer, i.e., a Staff Judge Advocate (SJA) in deciding the set of legal actions from the applicable rules after careful consideration of the cases identified by it. The approach, implemented as a software suite of tools, provides a repository of OPLAW/ROE sets of rules from different law sources. The repository allows for an information foraging and compiling capability. The selected sets of rules are then analyzed by the RULER tool for consistency and redundancy. Finally, a reasoning engine provides an "If-then' analysis capability to a user for the final selection of applicable rules in given mission situation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This is the final technical report of the contract entitled "Scalable Adaptive Architectures for Maritime Operations Center Command and Control" that was conducted by George Mason University between January 16, 2008 and April 30, 2011. The effort was driven by the Maritime Operations Center (MOC) operational concept (CONOPS[1]); its key idea is that these command centers will be scalable, using augmentation to morph from providing a basic capability for routine operations to conducting the more elaborate and complex operations needed to serve a Joint Force Maritime Component Commander or to become the headquarters for a Joint Task Force Commander. The research focused on discovering the overarching guidelines, techniques, and measures for achieving MOC scalability through augmentation. Some of the key issues explored were: (a) determining the best way to divide responsibility afloat and ashore in rapidly evolving situations; (b) synchronization and deconfliction of different battle rhythms, particularly between distributed staff elements and higher and lower echelons; and (c) dealing with multiple Rules of Engagement (ROE) across Areas of Responsibility (each may have its own ROEs that have to be deconflicted). Such a situation affects the choices that a MOC can make in selecting Courses of Action. The capabilities of the Net Centric Environment that are essential to effective collaboration among distributed staff elements must be considered as constraints in organization structure selection because they can affect the performance of the organization.

1.1 Scope of Work

The scope of work contains six research tasks plus an outreach task. Originally, the SOW contained five tasks; the sixth task was an extension of the original scope. As part of this task, work was conducted to develop a methodology for verification of rules, selected from across different Maritime Laws that are applicable in a certain given situation, against properties such as consistency, completeness, and redundancy. The approach is designed to assist a Staff Judge Advocate (SJA) in evaluating whether a proposed set of actions is not prohibited by the applicable rule set after careful consideration of possible inconsistencies that may appear in the relevant rules. The identification of such issues will help a SJA in developing course of action alternatives that are consistent with Operational Laws (OPLAWS) and Rules of Engagement (ROE). The final scope of work is summarized in the following task statements:

Task 1. Implement on a common testbed the algorithms for the design of command center organizations: fixed structures, variable structures, hierarchical structures and morphing strategies.

Task 2. Conduct literature research into command center augmentation strategies and use them as a starting point for developing candidate strategies and procedures for the MOC.

Task 3. Integrate the RULER tool and the Temper tool so that Rules of Engagement, security constraints and temporal constraints can be included in the MOC augmentation considerations.

Task 4. Develop scenarios for evaluating and testing the performance of alternative organizational designs. Conduct computational experiments to discover the underlying principles involved in augmentation planning and implementation.

[1] Maritime Headquarters with Maritime Operations Center Concept of Operations (CONOPS). US Navy Fleet Command, March 13, 2007.

Task 5. Integrate the System Effectiveness Analysis (SEA) Tool in the testbed and conduct trade-off studies for alternative distributed and deployed designs when MOC elements are ashore and afloat.

Task 6: Develop and implement a methodology for the management, verification, and visualization of Operational Laws (OPLAW) and Rules of Engagement (ROE) used by a Staff Judge Advocate (SJA).

Task 7. Conduct an outreach program.

Each task was completed and as a result both overall objectives were satisfied.

1.2 Research Accomplishments

Previous research efforts by GMU had developed the theory supporting a set of tools that could be used to design command and control organizations, but those tools were written in older programming languages, so they needed to be upgraded and modernized. The selected testbed for this research was the C2 Wind Tunnel (C2 WT), but it had not been populated with the tools needed for this effort. Ruler and Temper were immature tools and the SEA tool was a stand-alone application. Second Fleet and Booz Allen Hamilton were starting to develop detailed operational concepts for the MOC, but the operational concepts for changing MOC organizations to support changes in missions were only beginning to be addressed. Scenarios had not been developed. In short, the CONOPS existed but the details of its implementation were yet to be developed. Furthermore, a set of theories, tools and techniques for the design of organizations had been developed in the past, but needed to be updated to support the issues of scalability for the MOC.

As the research proceeded, GMU first developed a fundamental concept of augmentation that can be thought of as a micro view of scalability. This was followed by the development of a macro view of the MOC design problem that focused on the different organizational levels that apply to the types of situations a MOC may be used in. Then GMU focused on the use of tools to create MOC designs. The CAESAR III tool was used to examine alternative MOC organizational designs based on the micro and macro concepts. Finally, a MOC architecture design process was developed and illustrated with a case study that was based on the DOD Architecture Framework (DODAF) for describing not only the MOC organization, but the services, systems, and networks that support the organization.

To address the dynamically changing rules of engagement of a maritime force as it crosses different geographical areas, GMU developed an approach that uses a formal model checking methodology to detect redundant and inconsistent cases in a set of rules selected from across different Maritime Laws. The approach can provide considerable help to a Staff Judge Advocate (SJA) in deciding the set of legal actions from the applicable rules after careful consideration of the cases identified by it. The approach, implemented as a software suite of tools, provides a repository of OPLAW/ROE sets of rules from different law sources. The repository allows for an information foraging and compiling capability. The selected sets of rules are then analyzed by the RULER tool for consistency and redundancy. Finally, a reasoning engine provides an "If-then' analysis capability to a user for the final selection of applicable rules in given mission situation.

2. MARITIME OPERATIONS CENTER AUGMENTATION

2.1 Augmentation Strategies

In order to be able to design scalable MOCs, it is essential to have developed an operational concept for the MOC and the process for changing its scale. To develop credible operational concepts, research into augmentation strategies, as well as interactions with organizations that were developing MOC designs based on the CONOPS, was undertaken. Several interactions occurred with Second Fleet and Booz Allen Hamilton on their development of augmentation concepts for the MOC, and GMU used those concepts to inform the strategies it developed for MOC augmentation. The Booze Allen Hamilton document "Maritime Headquarters with MOC Core Processes Smart Pack"[2] provided a detailed description of the MOC design providing workflow descriptions of key MOC processes and the organizational elements that will carry them out. The Smart Pack used Business Process Modeling Notation as a method of describing the process (or activities) that will be performed by the various organizations of the MOC and the process flows between those activities and organizations in a set of task graphs. In 2010, the GMU developed descriptions of the MOC processes and organizational relationships were provided to RADM David L. Philman for review by the OPNAV N88 staff. They agreed that the representation provides a very good baseline from which to develop reconfiguration options.

Based on this research and collaboration with Second Fleet and Booze Allen Hamilton, a definition and a classification of organizational augmentation were developed. As part of this classification, two types of augmentations were identified, augmentation by staff or personnel (Type 1) and augmentation by function (Type 2). In the first case, augmentation is limited to the expansion of the staffing of existing organizational elements with no significant change in the functionality (tasks) supporting the MOC processes. In the second case, augmentation occurs through activation of functional elements (tasks) required by changes in the mission and in the prevailing ROEs.

2.2 Micro View of Augmentation

To better understand these two concepts and the migration from one configuration to another GMU developed the State Machine Diagram of Fig. 1. It shows four states of a MOC conducting command and control operations. There is a starting state for a MOC called the Baseline Operations that represents a MOC doing basic operations. From the baseline state, the MOC can go through a series of transitions as it scales up and down (augments and de-augments). If it just increases the staff/resources but still carries out the same operational activities and processes, it transitions to the Staff-augmented Operations state. In that state, it can continue to add or subtract staff resources based on augmentation decision events. If, while in the baseline state, the MOC needs to augment (add to) the operational activities (and it does not need additional staff) it can transition to the Functional Augmentation operations state. If it has already augmented the staff/resources and now adds one or more operational activities, the MOC transitions to the Fully Augmented State with both added staff and new operational activities. The MOC can progressively de-augment from the augmented states when the situation changes, as shown in the model.

[2] MHQ with MOC Core Processes SmartPack v. 3.0, 23 January 2008, Pre Decisional Draft by Booze Allen Hamilton.

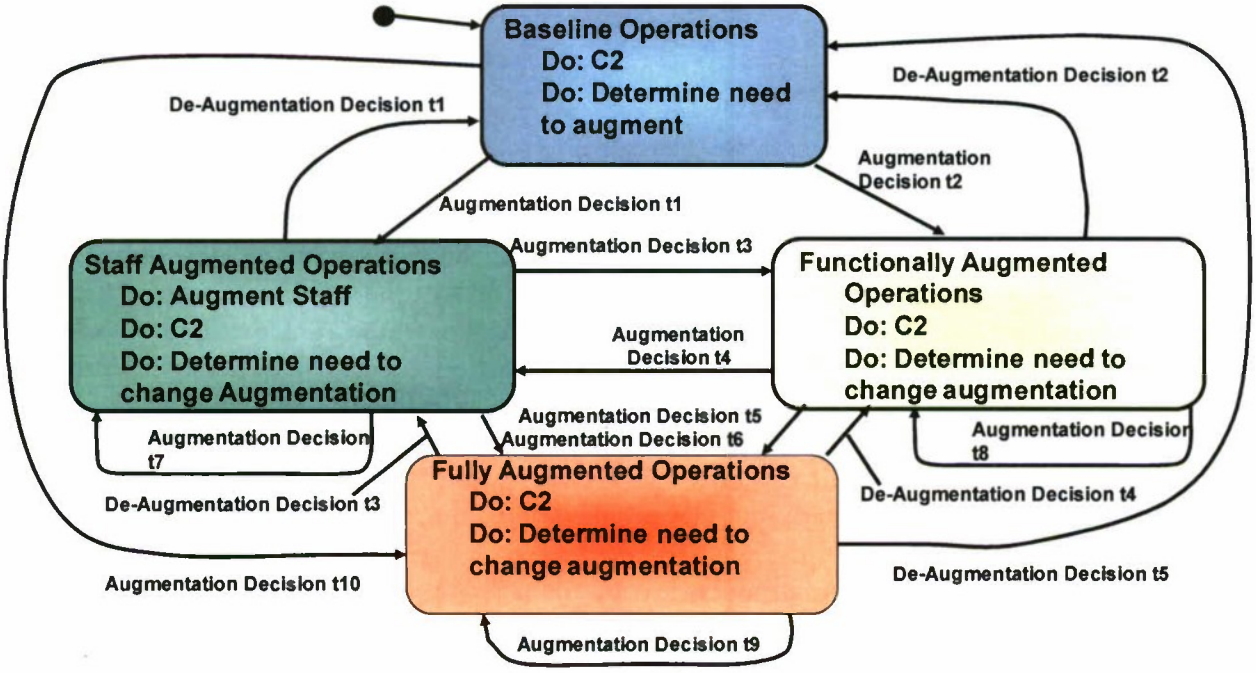


Fig. 1. State Machine Diagram of MOC Augmentation

GMU examined these concepts and their implications by using a simple example. We assume that a MOC is operating following a specific task graph with a set of staff that performs the tasks. It is assumed that each staff element is assigned to perform particular tasks in the task graph. This implies an allocation of the functions (Tasks) to the resources (staff). Figure 2 shows an example of a task graph with an allocation to specific members of the staff. These staff members are labeled "DM" for Decision Maker in the figure.

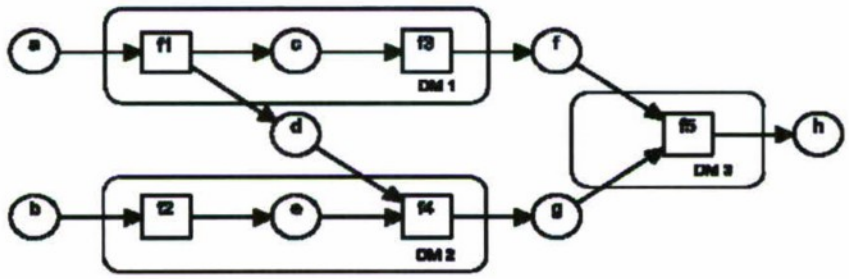


Fig. 2. Basic Task Graph Allocation

Now we assume that Decision Maker 2 (DM2) has become overloaded, and we need to bring in an additional staff element (DM 4) to support DM 2. There are several configurations that affect the information exchanges and coordination that enable this augmentation. These include DM 4 taking over either f2 or f4, DM2 and DM4 both working on f2 and f4 in parallel, or DM4 assisting DM2 in one or both of the tasks. These possibilities require different information exchanges and possible coordination. Figure 3 shows several of these. Figure 4 shows an example of DM4 assisting DM2 with function 4.

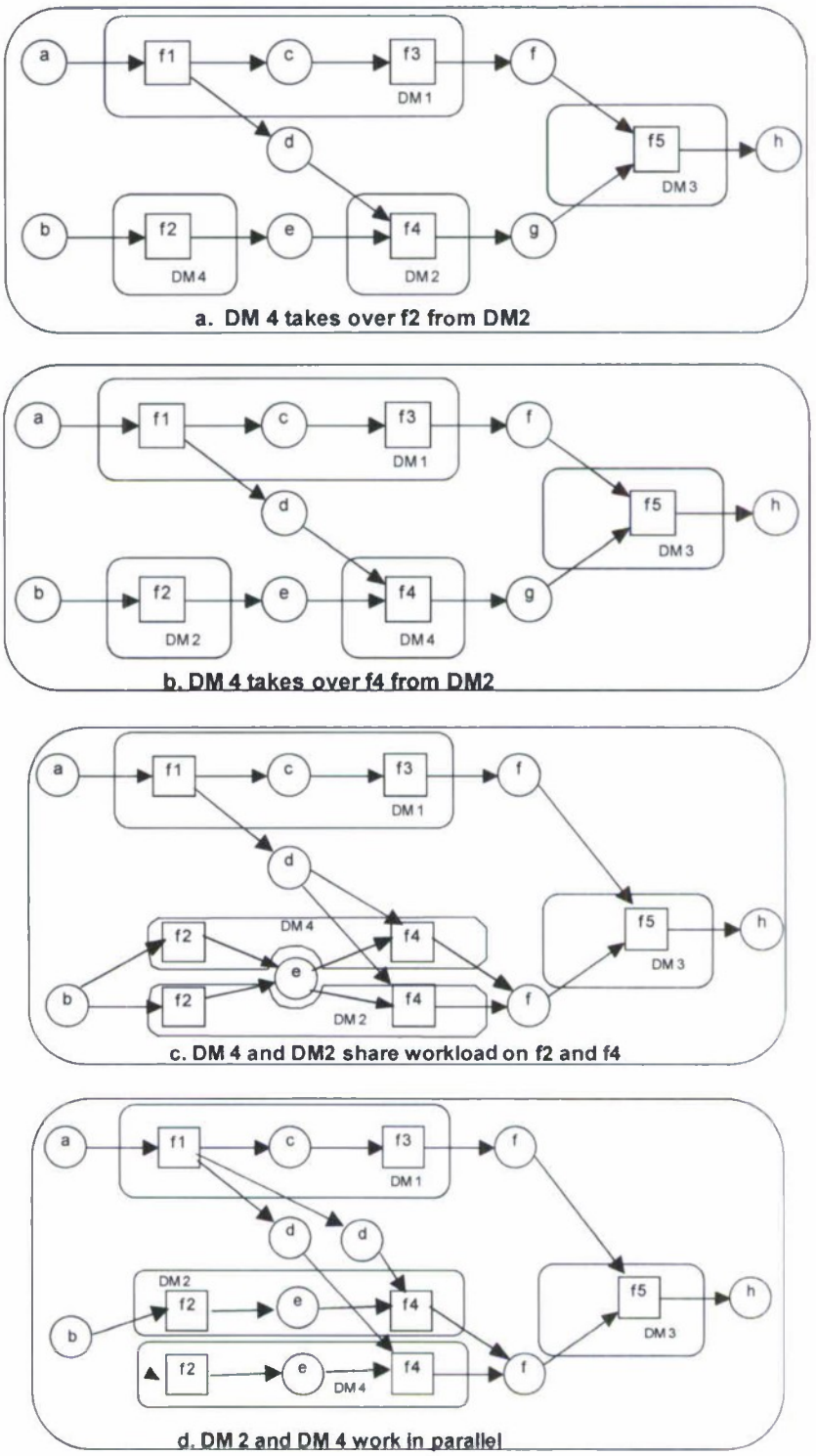


Fig. 3 a, b, c and d. Staff Augmentation Examples

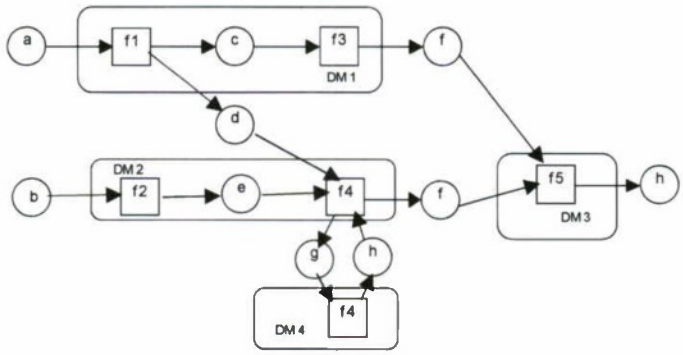


Fig. 4. Staff Augmentation with New DM4 Assisting DM2

Augmentation by function occurs when a MOC organization must carry out one or more additional functions or tasks that cause a change in the task graphs, but the MOC does not need to add additional staff to do this. Consider the case of adding two additional tasks (f6 and f7) to the original task graph of Fig. 2. Assume that these additional tasks can be completed by the original three staff members (DMs) by reallocation. Figure 5 shows an example with the new tasks being added to the existing task graph and the three original DMs re-allocated to perform them. If the original tasks cannot be reallocated, it may be necessary to bring in additional staff. Figure 6 shows an example where the original allocation is maintained and a new staff element (DM5) is added to the MOC to carry out the two additional functions. This results in a combination of augmentation by staff and function.

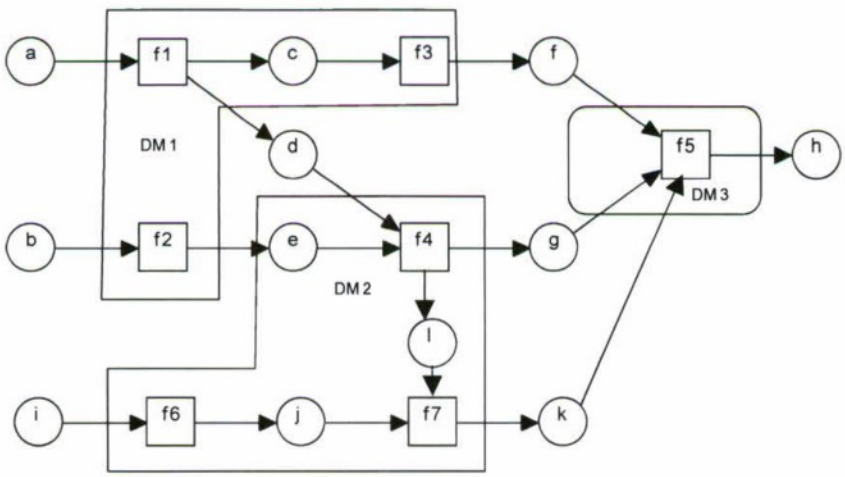


Fig. 5. Example of Augmentation by Function with No Change in Staff

These examples show that there are several factors to consider when the MOC must change scale. The existing and expected workload of the task graphs is considered a given, and it is assumed that personnel with the proper training and skills are available. As was illustrated in Figs. 2 through 6, there can be many ways the allocation of the tasks to staff could be carried out. Factors affecting the final decision include the amount of coordination that will be required between the staff members; more coordination and collaboration will increase the workload. Furthermore, the location of the staff has implications on the new MOC design. This is because

the communications infrastructure needed to support information exchanges changes with the location of the staff. If the staff are collocated, information exchanges are simplified; if they are not, the communications requirements and availability affect the allocation decisions.

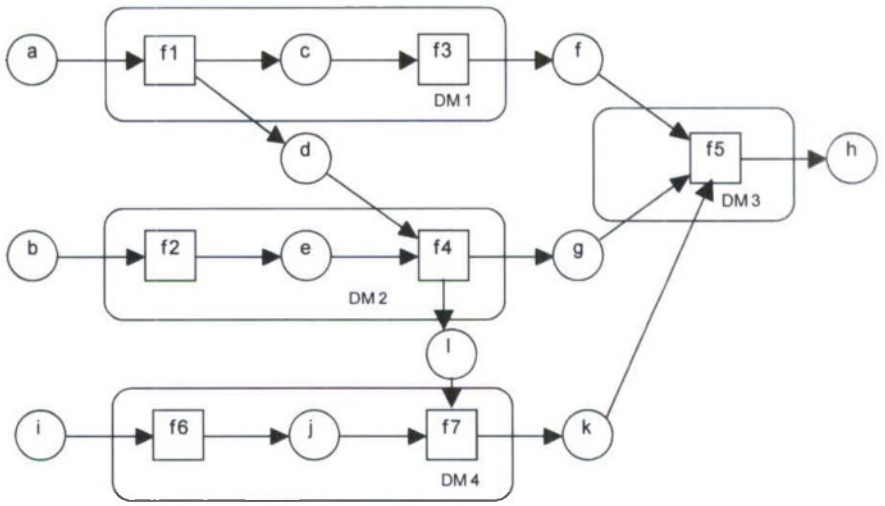


Fig. 6. Example of Augmentation by Function and Staff

2.3 Macro View of Augmentation

The above understanding of the basic or micro level concepts of augmentation by staff and function were supplemented by research on various augmentation approaches along with interactions with key organizations within the Navy. This research resulted in a macro view of MOC scalability using augmentation. Given a situation in which a Naval Task Force (NTF) commander is preparing for deployment or is already deployed and there is a change in mission, the MOC must design a new or changed organization to best meet potential mission needs for the period of the deployment. The MOC organization conducts Joint Operational Planning and Execution System (JOPES) processes[3] in support of the NTF commander for routine and potential missions during the deployment subject to the constraints of space, personnel, and connectivity anticipated for the potential missions. The new mission may include new tasks (functions) to be performed (and perhaps some old tasks may no longer be needed). This may result in a process model that is different from the original process model or it may result in the same set of functions with different information flows (affecting workload needed for some activities). The process (activity model) and battle rhythm will act as requirements for the augmentation strategy. Part of the design will involve finding the right resources (people, organizations, and systems with location) that will be added to the existing resources that satisfy those requirements. The design of a new MOC architecture including changing those processes and resources will need to be examined.

The NTF Commander has limited space on the command ship, and limited "overflow" line-of-site" workspace on vessels in the immediate action group. Additionally, some functional specialties are personnel limited and therefore cannot be made available for routine deployments.

[3] Joint Publication 5-0, Joint Operational Planning, 26 December 2006

The Commander's MOC satellite bandwidth is also limited, and is even further limited in a contested cyber environment.

Figures 7 a to e illustrate some of the possible MOC roles or missions. Figure 7a represents a Routine Operations MOC at the Operational Level with an Expeditionary Strike Group (ESG) performing at the Tactical level. Figure 7b shows a MOC of Large Naval Operations with the MOC supporting an Expeditionary Strike Force (ESF); ESG and Carrier Strike Group (CSG) commanders use smaller Operations Centers on their battle decks for tactical level command and control. Figure 7c shows a MOC supporting a Joint Force Maritime Component that is under a Joint Task Force (JTF). Figure 7d shows a MOC supporting a Maritime Component that is also acting as a JTF. Figure 7e show a MOC supporting a Commander of a JTF.

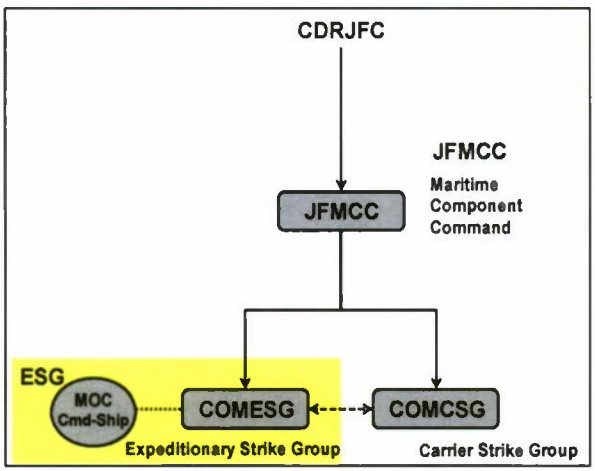


Fig. 7a. Routine Operations

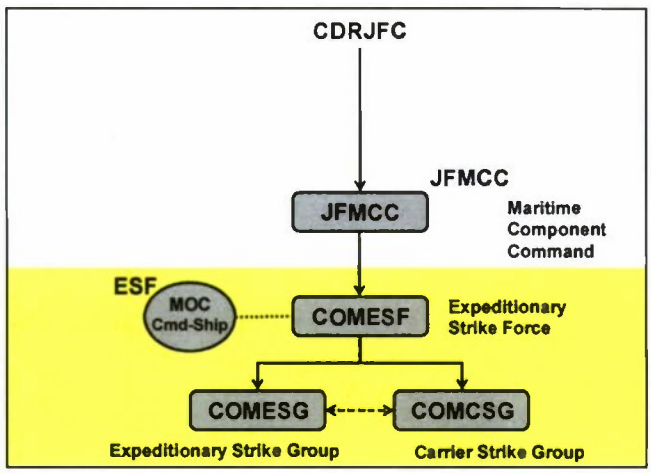


Fig. 7b. MOC for ESF

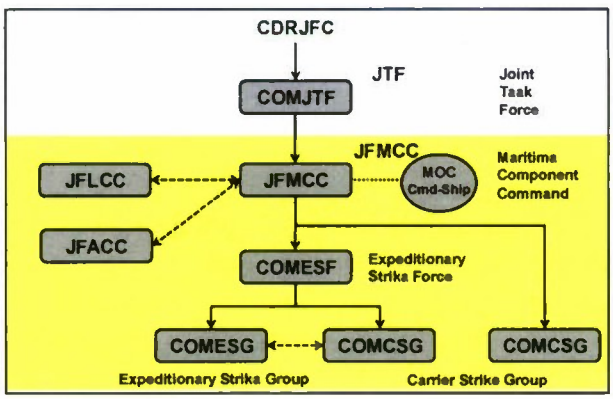


Fig. 7c. MOC for JFMCC

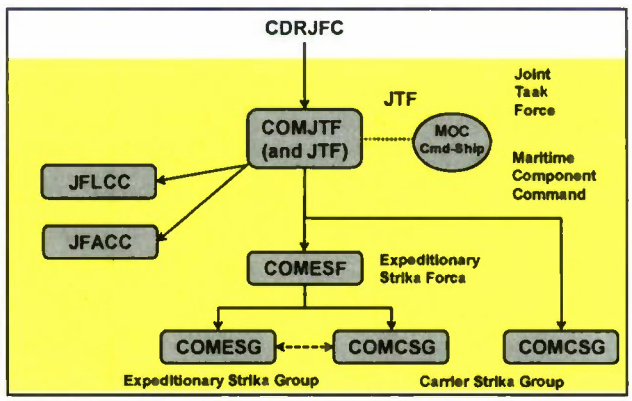


Fig. 7d. MOC for JFMCC/JTF

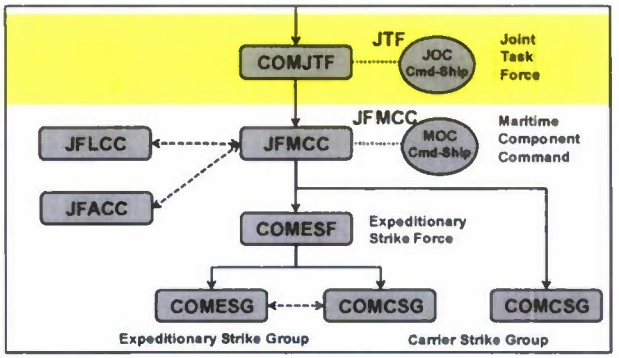


Fig. 7e. MOC Supporting a Joint Task Force

Our research was based on the following assumptions that apply when designing a MOC organization including the transition from the current design to the new design.

- The MOC will be configured to conduct missions commensurate with potential taskings to the deployed commander.
- The MOC will employ basic JOPES processes to conduct all missions.
- Basic MOC external relationships are consistent: superior commander, subordinate commanders, peer commanders, external agencies (see Figs. 8a and b).
- Subordinate commanders will make necessary expertise available.
- The MOC is constituted with expectation that it will plan and execute operational level operations and direct tactical level operations.
- The Commander will want to conduct "highest" level planning organically and will delegate other planning to subordinate commanders.
- Operations will be contested - the commander will organize to mitigate these by minimizing bandwidth requirements.
- USN (and DoD) will minimize the number of high demand/low density subject matter experts deployed in anticipation of requirements.

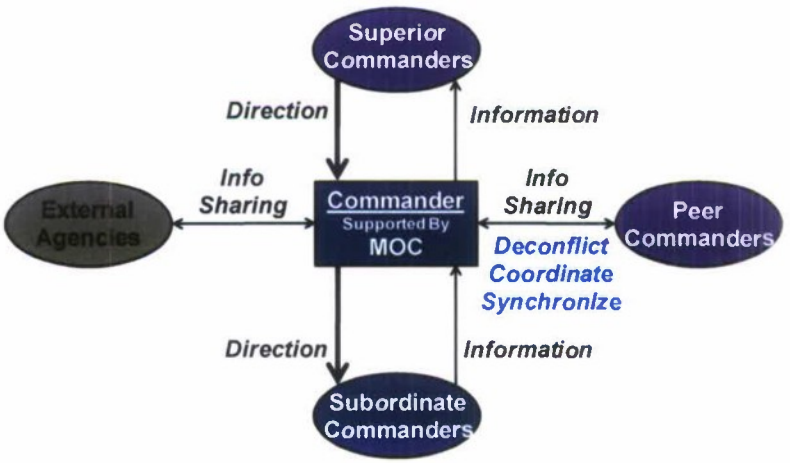


Fig. 8a. Key information Exchanges Between MOC and Other Organizations

MOC Role	Controls	Coordinates with
ESG/CSG MOC	ESG (or CSG)	Adjacent Strike Groups
ESF MOC	ESG, CSG	Adjacent Strike Forces, Groups
JFMCC MOC	ESF, Strike Groups	Functional Components
JFMCC (& small JTF) MOC	ESF, Strike Groups, JFCCs	Service Components
JTF JOC (Joint Ops Center)	JFCCs, Service Components	

Fig. 8b. MOC Information Exchanges Based on MOC Role

In addition to these assumptions, three constraints also must be considered. Each constraint can be mitigated as indicated.

(1) Bandwidth limitations (particularly in contested environments). This can be mitigated by minimizing the need for information transfer over contested links.

(2) Command platform workspace limitations, which can be mitigated by the transfer of workload to other platforms within line of sight or the workload can be accomplished through reach back (beyond line of sight).

(3) Insufficient expertise in some functional areas to meet all deployment requirements, which can be mitigated by arranging reach back support for high demand/low density functional expertise.

3. MOC ARCHITECTURE DESIGN APPROACH

3.1 DoDAF Compliant Architecture

With this understanding of the micro and maero views of the MOC augmentation eoneept, GMU developed an arehitecture design approach that would enhanee the ability to evaluate alternative MOC designs. The DODAF version 1.5 initially was ehosen as the framework for developing the representation of the arehiteeture description; the approaeh was later updated to the DODAF version 2.0. The Unified Modeling Language (UML) was selected as the architecture modeling language. To develop a design of a new or ehanging MOC, the approach uses a layered architecture with an Operational Viewpoint, a Systems Viewpoint, and possibly a Services Viewpoint. The Operational Viewpoint deseribes the MOC organizational structure, operational activities (tasks and task graphs) that are performed by those organizations, and information exchanges between those organizations. These elements of the Operation Viewpoint are defined by the JOPES proeess that will be used by the MOC and the organizational entities that will earry out the processes. Based on the CONOPS, MOC organizational entities as a minimum include an Intelligenee and Assessment Center, Future Plans Center, Future Operations Cell, and a Current Operations Cell. In the Operational Viewpoint the boundary of the MOC with its interfaces with other organizations is clearly defined as shown in Fig. 8a. Systems and Services Viewpoints will be created to describe the System Nodes, Systems, Services, Interfaces, and Communications Infrastructure that support the Operational Viewpoint. It should be noted that there are likely to be alternative Systems and Service Viewpoint descriptions that are capable of supporting the Operational Viewpoint. A mapping from OV to SVs is provided by the SV-5 and the SvcV-5 (operational activity to system funetion/serviee funetion traeeability matriees). Executable models of the OV and SVs can be created to examine behavior and performanee issues.

This approaeh assumes we know what the new proeess is, what the new organizational resourees will be, and how we will allocate the processes to those resources. Then, given these inputs, the analysis will show whieh organizational resourees do not need to ehange and whieh ones do because their task graphs will change. It also enables the architeet to examine the impact of different Systems and Service locations that will support the organizations on the overall behavior and performance of the new MOC. The architecture description depicts local and long haul communications links some of which may have bandwidth limitations and vulnerabilities to cyber exploits.

The process can be summarized as follows. *Given a new MOC mission that is described with a new process model, battle rhythms, and set of potential Person-type performers with specialty, availability, and location, design one or more new MOC architectures including Operational, Services, and Systems Viewpoints taking into account the changes needed to make the transition from the existing MOC structure to the new design.* The designs should be evaluated to ensure the ability of the new design to meet behavior and performance requirements (whieh must be provided as an input). This ean be done by estimating performanee or by converting the description to an executable model that can run in simulation mode to collect behavior and

performance data. The design should include the method of transition, that is, which performer resources and resource flows should be added and when.

To illustrate this process an example scenario was created. In the scenario, a certain MOC is responsible for ongoing operations in an Area of Responsibility (AOR). Its mission is to provide operational level command and control of a maritime operation to maintain sea lines of communications (SLOC) and monitor non-military sea based traffic including fishing and recreational vessels. It includes support to land based operations conducted by a Marine Expeditionary Unit that is part of an ESG in a county in the AOR. During the basic mission a natural disaster occurs on an island removed from the ongoing operation but within the AOR of the MOC. The island has significant US interest because of a communications station located there. The disaster has provided an opportunity for insurgents on the island to attempt a coup. The current US friendly government can no longer provide for its population and cannot guarantee security of the communication facility. The MOC is given responsibility for both the original operation plus the new operation that requires either a second Marine Expeditionary Unit or an increase in support to the primary ESG to support both humanitarian assistance operations and security operations for the communications facility in the new disaster area. The MOC will need to scale up in terms of activation of additional tasks and additional organizational resources to support the increase in responsibility.

Several sets of operational tasks are in an idle or dormant state and will have to be activated. For example, the Futures Plans Center tasks involved in (deliberate) operational planning are dormant having been completed for the basic mission. Also the Task Graph for Developing an Operational Environment and Sustainability Plan will need to be activated.

The MOC is located on a vessel associated with the Maintain SLOC mission in the AOR. It has connectivity to Higher Headquarters and the other organizations connected to the Global Information Grid as well as organic ISR assets needed to support the mission. The change in mission means additional capability must be incorporated in or provided to the MOC. This requires activation of several task graphs (processes), e.g., Conduct Operational Planning for the new tasking. It requires connectivity to the locations of the resources (systems and services with humans). The OV changes; new functionalities are activated and additional cells may be created. Several candidate System Views are created and evaluated so the "best" solution can be selected based on performance and vulnerability analyses.

Figure 9a shows a portion of a Task Graph for the original Future Plans Center (FPC) and Fig. 9b shows the Systems and Networks that support that workflow. The current MOC operation is small enough that all of the operational activities take place on the Command Ship and the systems that support those also are located on that ship. The MOC communicates with its higher headquarters via SATCOM and with the tactical units via Line-of-Site Radio.

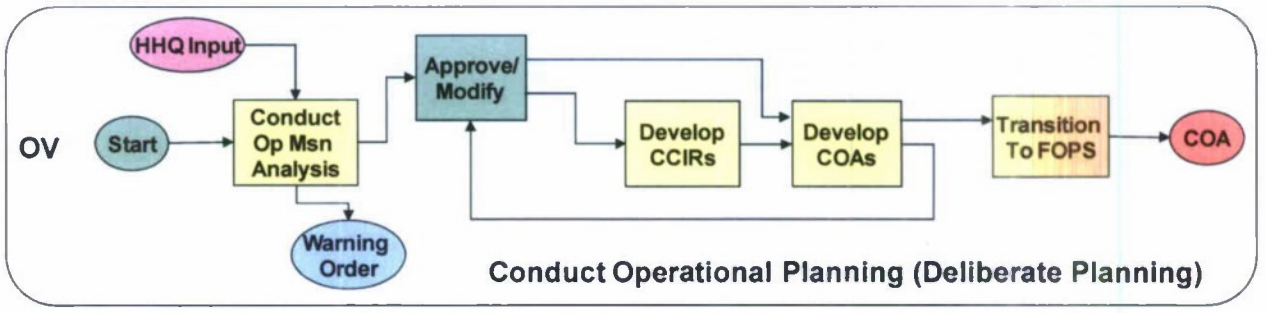


Fig. 9a. Original MOC Task Graphs for Future Plans

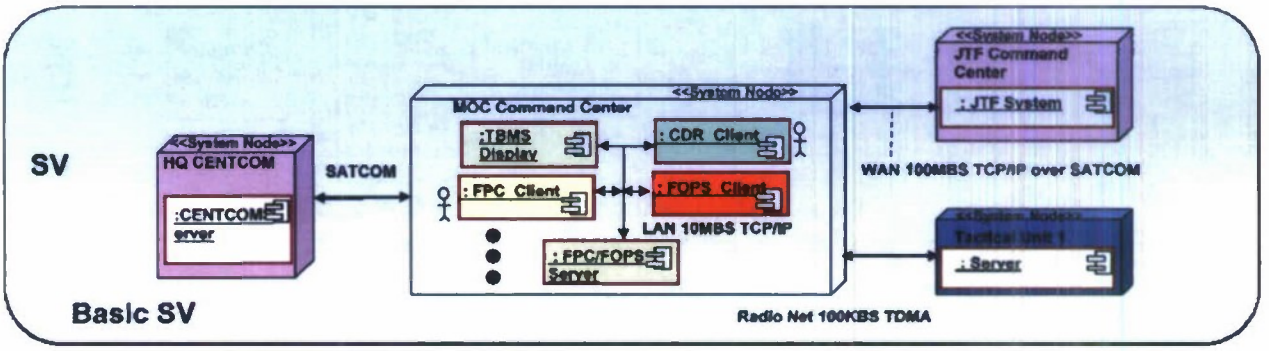


Fig. 9b. Corresponding Systems Viewpoint

Figure 10a shows a portion of the new MOC workflow (Operational Activities being performed by the MOC Future Plans and Future Operations organizational resources). The new mission will require not only the workflow for conducting mission analysis, issuing Warning Orders (WARNORD), developing Commander's Critical Information Requirements (CCIRs) and developing Courses of Action (COAs), it also must perform the new tasks to develop an Operational Environment Management Plan to deal with the new situation. Furthermore there will be a need to augment the FPC as it performs its more complex Operational Planning activities. This means added functions and added staff.

In the next step the architect must determine the new Systems Viewpoint. This will be determined in part by the location of the additional staff that will be performing the new Future Plans Center operational activities and the connectivity to those locations. Figure 10b shows two candidate SVs labeled Version 1 and Version 2 designed to support the increased workload to carry out Operational Level Planning. In Version 1 additional client systems are added to the MOC command ship to support the additional people that will be working in the FPC and the Futures Operations Cell (FOPS). This will only be viable if there is sufficient space and expertise on the Command ship. Version 2 shows the Future Plans Cell split with part of the tasks being accomplished on the Command ship and the rest at a reach back facility. In the example, this reach back is co-located with the HQ CENTCOM. The connection is via SATCOM. If the capability to have a reach-back cell at HQ CENTCOM already exists, this can be set up quickly.

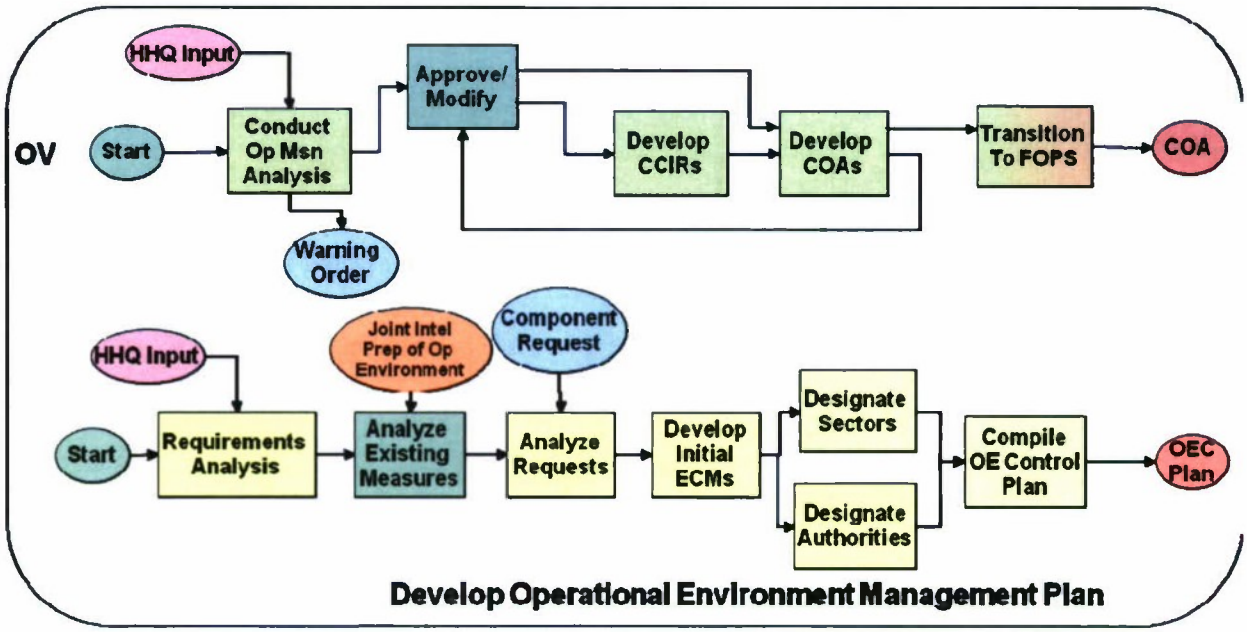


Fig. 10a. New Task Graphs for the new MOC Mission

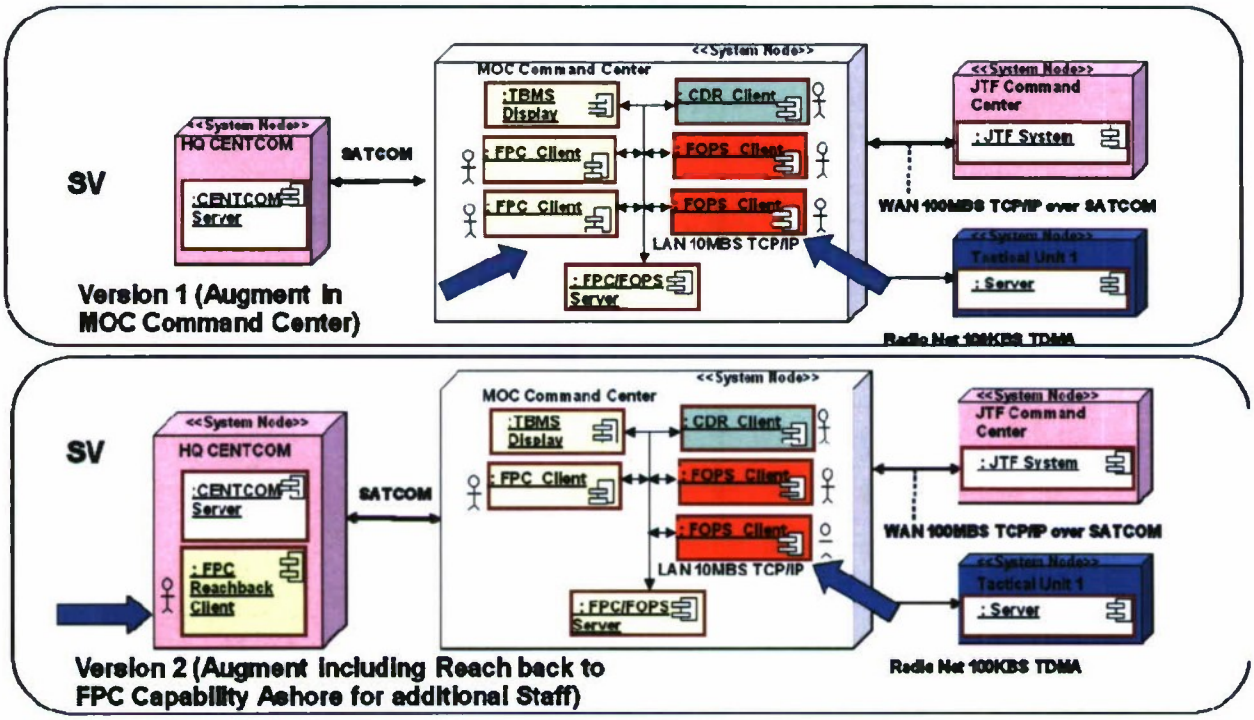


Fig. 10b. Alternative System Viewpoints for the MOC Planning Activities

The analysis is expanded to include the new functionality of developing the Operational Environment Management Plan, which will be accomplished by the FPC. Two options are considered, one with all functionality provided on the Command Ship and the other using reach back. The need to interoperate with the MOC Intelligence and Analysis Center is also shown, with this capability also residing on the Command Ship. Figures 11a and b show these alternative

System Viewpoints. Another perspective of these alternatives is shown in Fig. 12 which foeuses on the geo-location of the various capabilities that can be interconnected to enable the new MOC to operate. The deeision as to whieh eonfiguration to use will be based on several faetors ineluding the available space on the Command Ship, the loeation of capabilities to support the MOC, the expertise of the staff at those locations, the ability of the communications links to support the required information and data exehanges, and the potential risks of disruption if the cyber environment becomes contested.

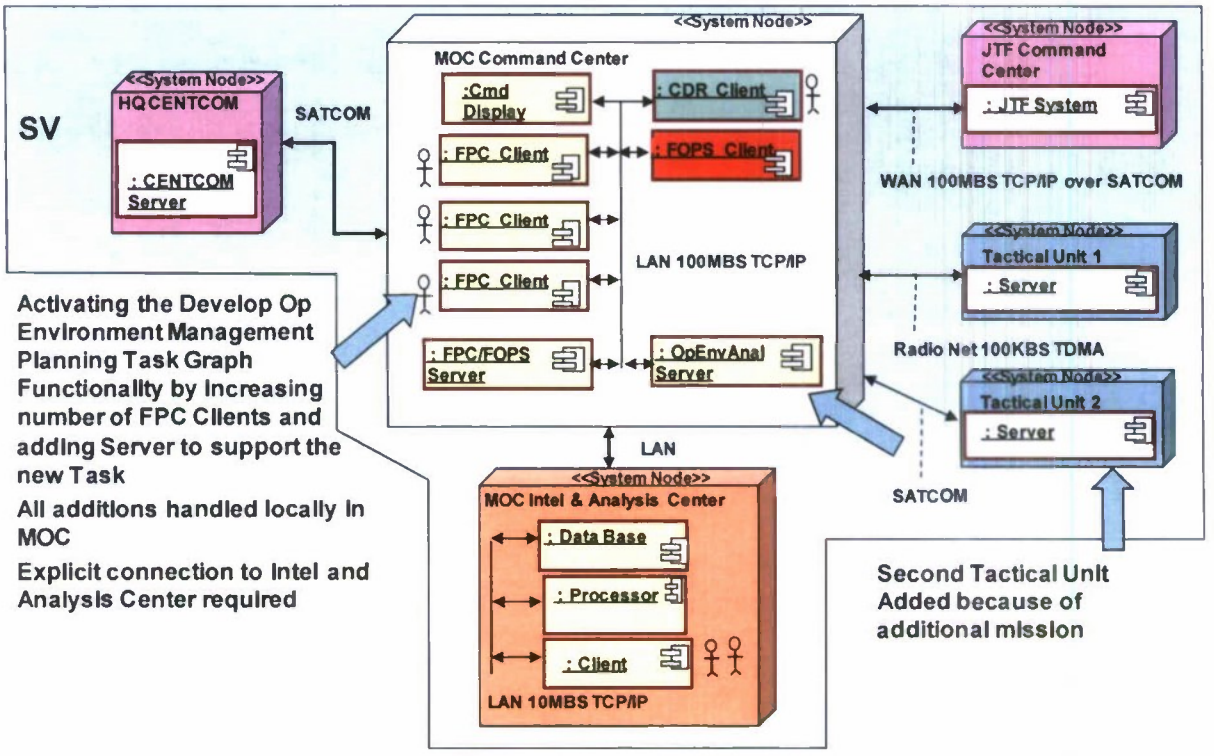


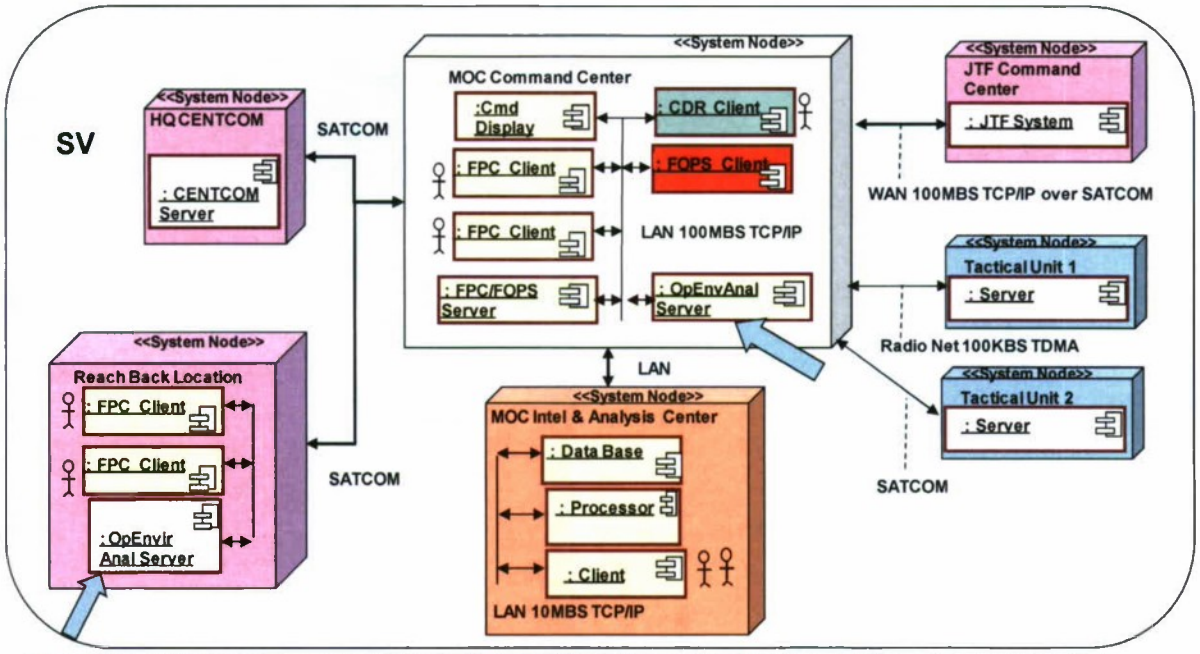
Fig. 11a Systems Viewpoint with all Capability on Command Ship

3.2 Tool and Process Support for Developing Candidate MOC Architectures

The analysis described in the previous section indicates that there is a lot of latitude in designing an arehiteeture for a MOC that is ehanging its seale ineluding the sequenee of ehanges needed to transition from an existing MOC eonfiguration to a new eonfiguration to support a ehange in mission. In order to provide assistanee to an arehiteet, GMU experimented with the tool called CAESAR III to develop proeesses for examining augmentation by staff and augmentation by funetion. The tool ean assist in the design of the organization by using the lattiee algorithm developed in previous ONR funded researeh. It is based on the five stage deeision maker model. This model can be used to model a team of individuals or a team of deeision making entities sueh as an organization made up of eenters and eells.

The first step in developing the design is to identify the deeision makers (organizational entities) and then the types of interaetions that take plaee between them. This is aeeomplished by using the task graph for the new MOC mission and alloeating the tasks to the decision making

organizations as was discussed in the description of the micro view of augmentation and the Operational Viewpoint of the DODAF compliant architecture description. The model also contains connections from a single node presenting all external information sources and a second node representing all information sinks.



- **This Functional Augmentation is enabled by using a Reach Back Location to extend the functionality of the Future Plans Center (FPC). Second OpEnvAnal Server included in MOC Command Center**

Fig. 11b. Systems Viewpoint with some Capability at Reach Back Location

Continuing with the example, an organizational model of the existing MOC was created in CAESAR III as shown in Fig. 13. This model shows the main organizations of the current MOC, namely the Intelligence and Assessment Center (which has been divided into two parts), the Future Plans Center, Future Operations Cell, the Current Operations Cell, and the Commander, who approves all plans and directives. The model shows the main flow of operational information between the cells as the MOC performs the operational level planning and execution monitoring functions. In addition, it models the fact that in the current MOC the links from the Command to the Current Plans and Current Operations organizations are not always used. Those links are created as variable links. The analyst uses the CAESAR III tool to run the lattice algorithm on the design. It generates the lattice shown in Fig. 14 that has four feasible solutions including one Mino (Minimally Connected Organizations) and one Maxo (Maximally Connected Organization). The tool can also generate Petri net models of selected organizations. The Petri net of Fig. 15 represents the Maxo (Maximally Connected Organization). One interpretation of this can be that the workload on the Current Plans (DM 5) and the Current Ops (DM 6) organization may be increased when those variable links are active because they must process the addition inputs.

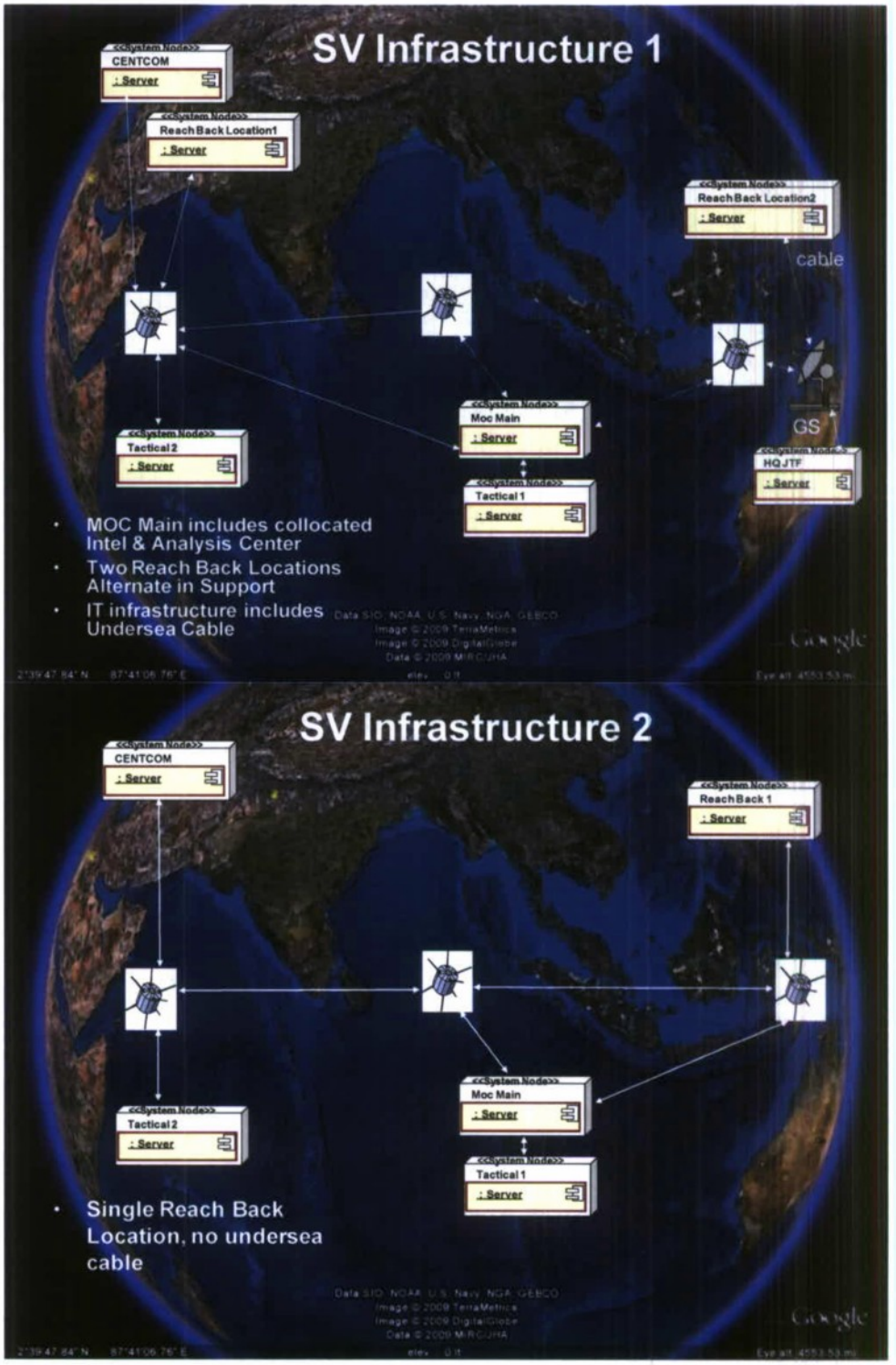


Fig. 12. Geographical Representation of Two Alternative System Viewpoints

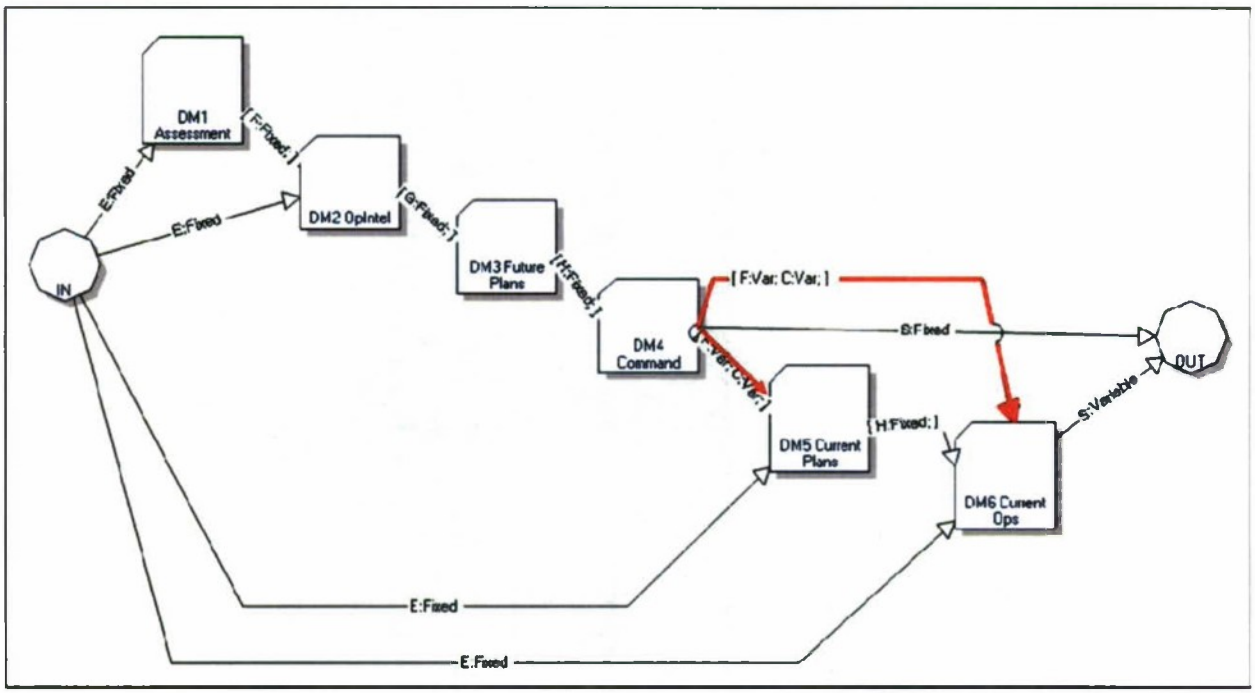


Fig. 13. Current MOC Organization Represented in CAESAR III

The lattice algorithm indicates there are four feasible organizations as shown in the lattice of Fig. 15. The Petri net of Fig. 14 represents the Maxo (Maximally Connected Organization). One interpretation of this can be that the workload on the Current Plans (DM 5) and the Current Ops (DM 6) organization may be increased when those variable links are active because they must process the addition inputs.

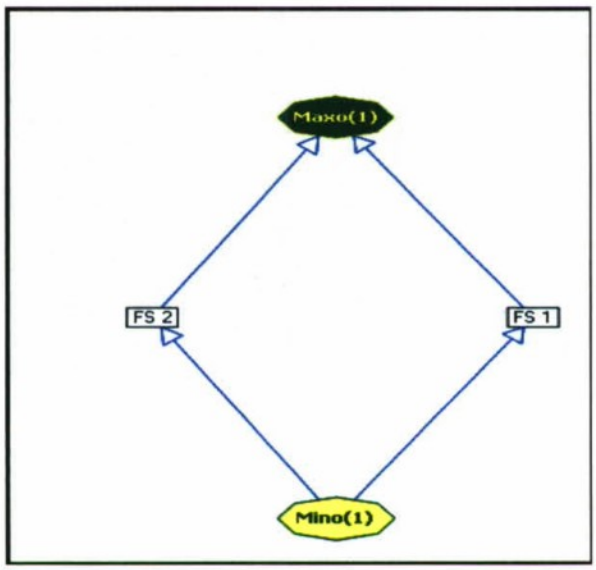


Fig. 14. Lattice of Feasible Organizations of the Original MOC

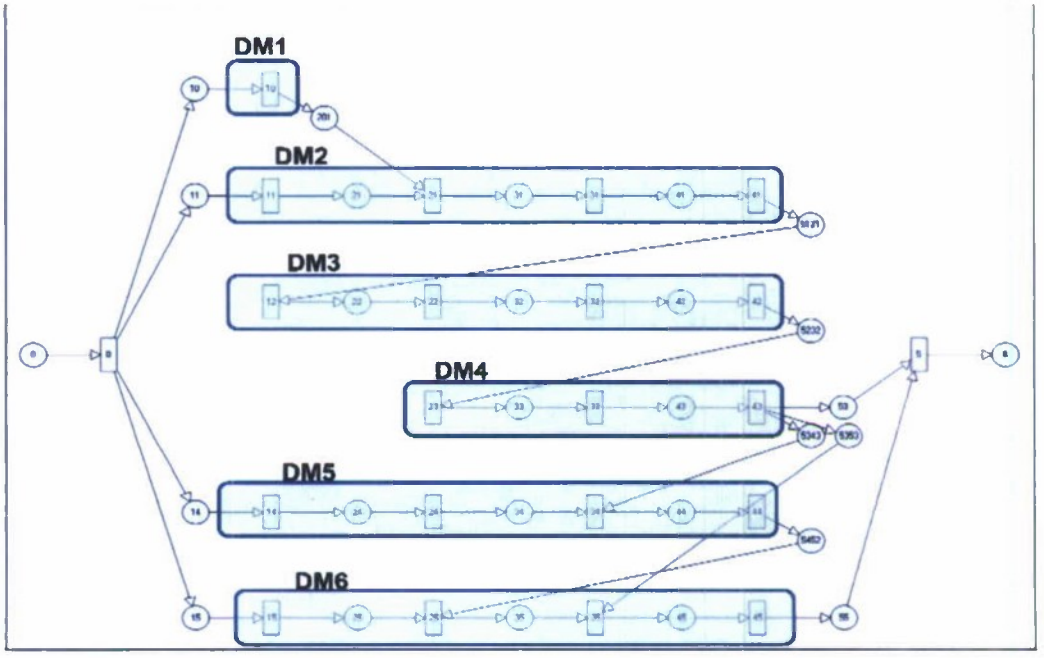


Fig. 15. CPN model of Original MOC Organization

Assume that the increase in mission requirements results in the need to provide staff augmentation to the Operational Intelligence portion of the Intelligence and Assessment Center and staff augmentation of the Future Plans organization. This can be represented in CAESAR III by adding two Decision Making Organizations that will be carrying out the same functions as the original organization, but sharing the workload. Figure 16 shows the CAESAR III representation of the staff augmented MOC. In this design, the two additional Decision Making organizations are added representing the additional staff supporting the MOC intelligence functions and the Future Plans functions. As was described in the discussion of the micro aspects of augmentation, there are several ways this augmentation can occur. One design question is whether these two additional entities should work strictly in parallel with the original organizations or whether there should be interaction between the augmentees and their original organization counterparts. If there is no interaction, then the location of the augmenting staff is not constrained except by communications capability. But if there will be interaction between the original organization and the augmentees, then the increased coordination can increase the workload of the organizations and add additional requirements for communications links if the augmentees are not collocated with their original counter parts. Variable links are used in the CAESAR III model to include these options in the design as shown in red in Fig. 16.

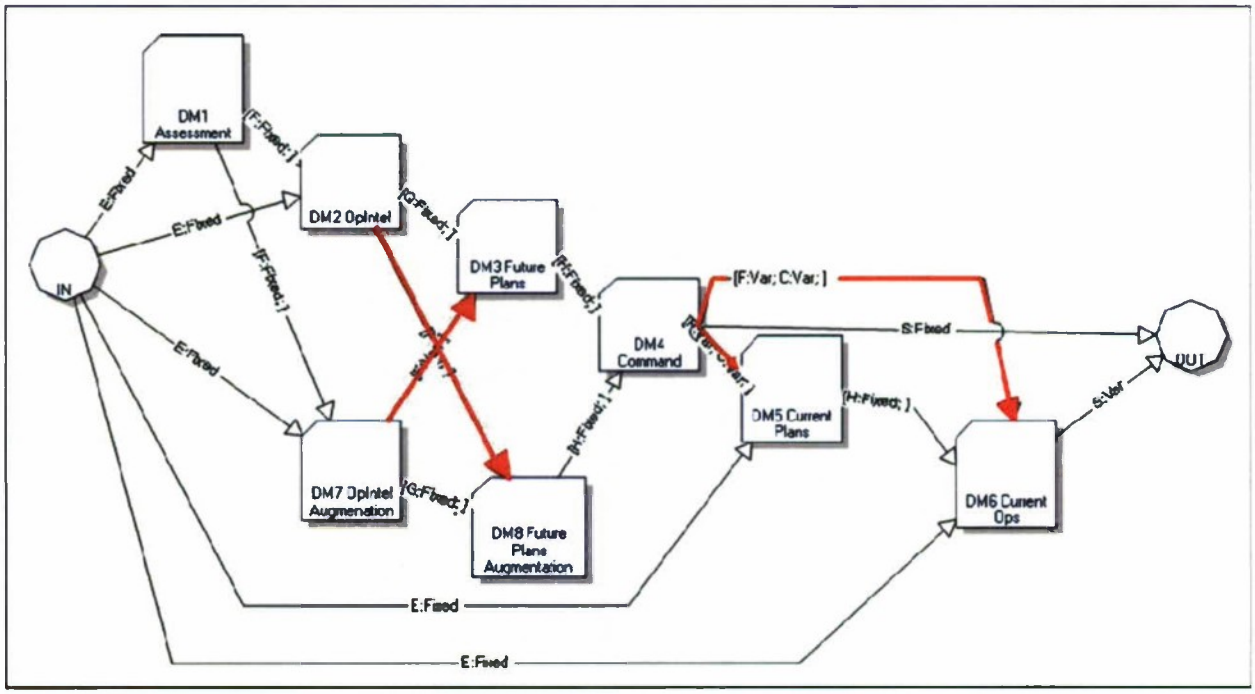


Fig. 16. Staff Augmented MOC with Variable Links

Running the Lattice Algorithm with this design results in the larger lattice shown in Fig. 17. It shows 14 feasible organizations with one Maxo and one Mino. The Mino and the three organizations that are close to it (FS 1, FS 2, and FS 9) represent the organizations where the augmentees operate in a pure parallel situation. The four organizations have the same interconnections as the four shown in the original MOC design with the variable links between the Command and the Current Plans and Current Ops plus new information flows between the two augmentation cells. In the remaining organizations in the lattice, interaction takes place between the original OPINTEL organizations and the augmented Future Plans and also between the augmented OPINTEL organization and the original Future Plans. Figure 18 shows the Petri net of the minimally connected organization, which has pure parallel augmentation on the left and the Petri net of the maximally connect organization on the right which requires the most coordination as well as the ability to set up communications between augmentees and their counterparts. The links that represent the coordination and the possible need of additional communications capability are shown in red. The organizations below the MAXO require fewer interactions and therefore may require less workload and less communications infrastructure. In short, this analysis gives a good indication of the options available organizationally. The architect can down select from this set of organizations by computing workload for each design, determining the subset that doesn't exceed workload capabilities of the organizational entities and then examining the communications requirement based on the location of the various organizations as was described earlier when using the Systems Viewpoint.

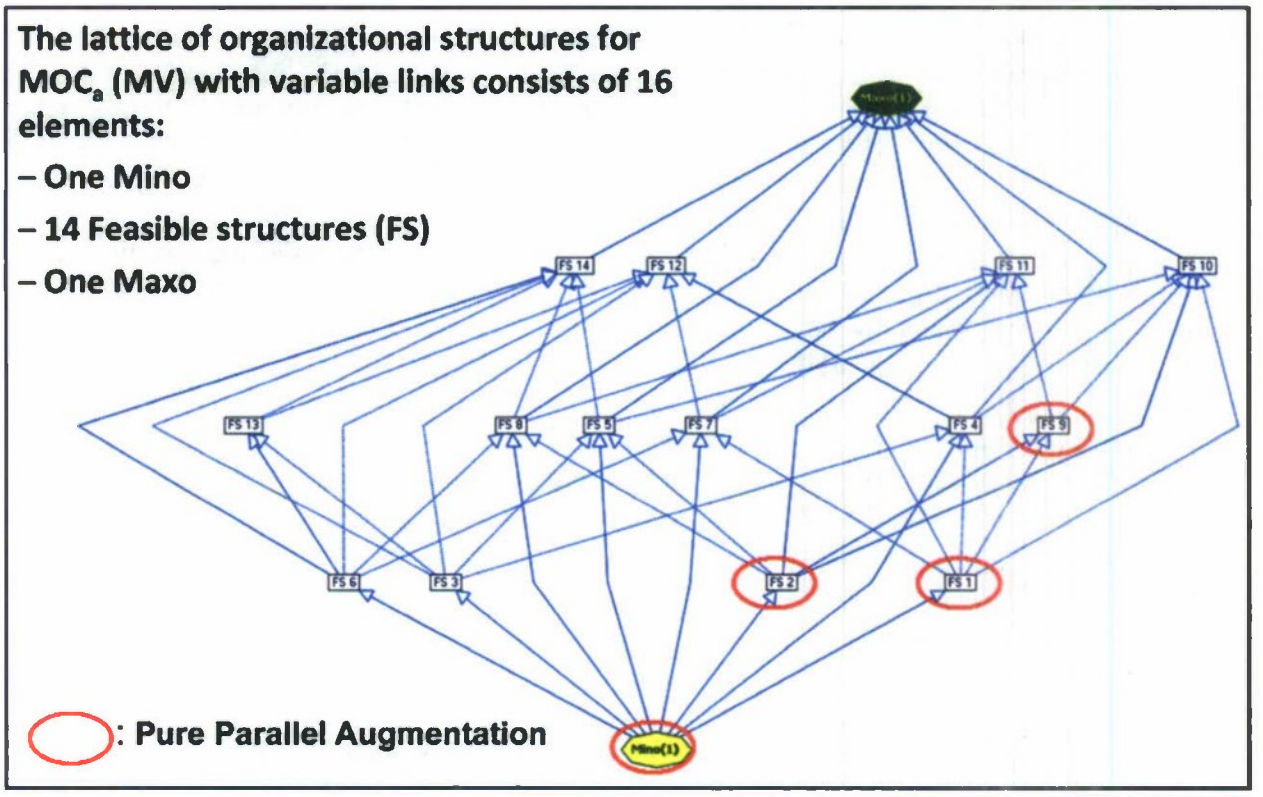


Fig. 17. Lattice for the Staff Augmented MOC

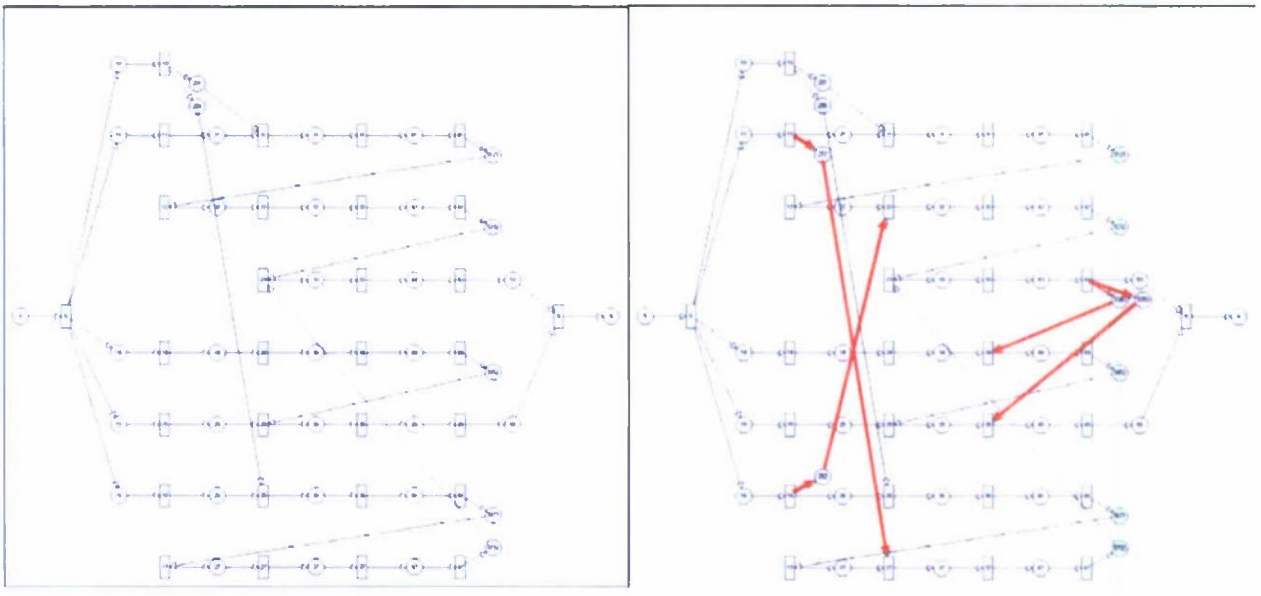


Fig. 18. The Petri Nets of the MINO and MAXO of the Staff Augmented MOC

GMU also examined the use of CAESAR III for the analysis and design of Augmentation by Function. Assume a MOC is performing a set of functions and a change in mission causes it to need to add one or more functional capabilities (functions). The issues with adding functions are similar to adding personnel (augmentation by staff): what are the information exchanges required to perform the additional functions (which may be described in a DODAF Operational Viewpoint), which organizations will perform those functions and where will they be located. In our example, we assumed that the MOC has a Command Vessel where MOC organizations exist and perform the basic operational activities (task graphs). There is limited space on the Command Vessel so it may not be possible to perform all functions on it, particularly new ones. The MOC may be able to use additional vessels under MOC control to handle the added MOC functions although they also have space constraints. Communications with these vessels is usually by Line of Sight, so bandwidth and the more common cyber vulnerabilities are less of a factor. Adding new functions also may be accomplished by using a reach back capability to organizations what have required expertise. Usually there is no constraint on space for the reach back organizations, but the long haul communications required limits bandwidth and has increased vulnerability issues.

This leads to the conclusion that there are seven different configurations that can be used to support MOC functions as shown in Fig. 19.

	Location		
Option	Command Vessel	Line of Sight Vessel	Reach Back
1	All work done here		
2		All work done here	
3			All work done here
4	Partial work done	Supporting Command Vessel	
5	Partial work done		Supporting Command Vessel
6	Partial work done	Supporting Command Vessel	Supporting Command Vessel
7		Partial work done	Supporting LOS Vessel

Fig. 19. Options for Selecting Locations for Performing Functions

3.3 Architecture Evaluation

To examine the use of CAESAR III to support the design and evaluation of alternative architectures for augmentation by function, the basic scenario was elaborated and an example created that focused on a MOC Future Operations Cell. It is assumed that in the original MOC configuration, the FOPs Cell is performing according to the FOPs task graph. The FOPs divides responsibility for different operational functions amongst various functional planners. When the MOC mission changes, one or more additional operational functions may be required to be handled by the FOPs cell. For example, if the change in mission now requires performing the activities involved in developing a Mobility and Sustainability Plan, the FOPs cell will need to perform these additional activities (functions). The Task Graph for the FOPs Cell can be represented as shown in Fig. 20. In this task graph, the FOP cell receives guidance and uses it to divide the work into different functional planning areas. Developing the new Mobility and Sustainability Plan is a new function that must be performed. As the planning functions are

completed, the outputs are assembled in a single Integration Function. The Integration Function then releases the planning results (Asset Allocation decisions).

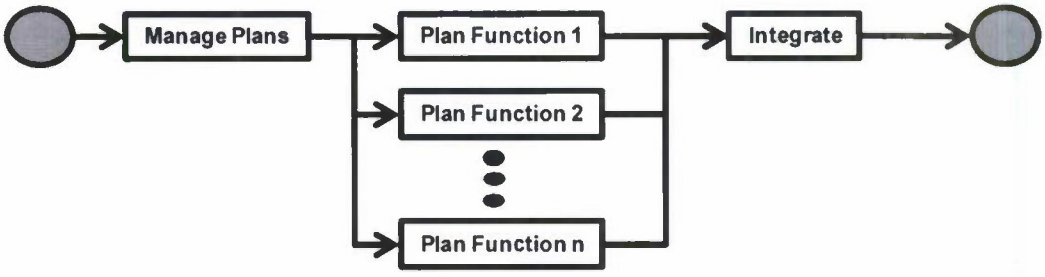


Fig. 20. Notional FOPS Task Graph

Each task or function is carried out by an element of the organization. The information exchanges between the organizational elements can be described as shown in Fig. 21. A task allocation table can be created that show these allocations. The organization design problem is constrained by the availability of people will the right skills and by the amount of space available in various locations, e.g. the Command Ship, Line of Sight Vessels, and reach back locations.

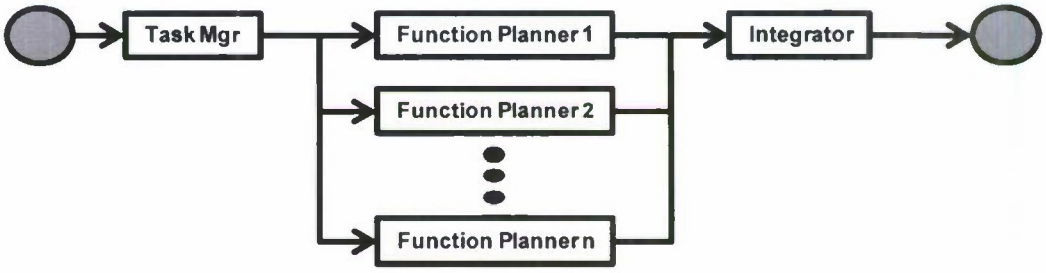


Fig. 21. FOPS Organizational Relationships

A model of organization can be created in CAESAR III as shown in Fig. 22. We assume that there are three functional planning organizations in the new MOC. The model was created using the following assumptions. The Task Manager is located on the Command Ship and sends the appropriate guidance to each functional planner. Each planning function will be initiated by a functional planner organization located on the Command Ship. In the example there are three such functional planners labeled DM 1, DM 5, and DM 8. Each carries out the task graph for planning the assigned function. We assume that DM 8 is responsible for the new task of Developing the Mobility and Sustainability Plan. The other planners are optional planners located on a Line of Sight vessel or at a reach back location who could support the planners on the command ship. Because these planners are part of the trade-off analysis, the links between them and the Planner they support on the Command ship are shown as variable links. The lattice algorithm is run in CAESAR III and the resulting lattice is shown in Fig. 23. The lattice has 24 feasible solutions with one Mino and one Maxo. The Mino is the organizational structure with all the functions performed on the Command Ship. The Maxo uses all of the organizational options of the design. This includes Option 6 from Fig. 19 for Functional Planning areas 1 and 2 and Option 4 for Functional Planning Area 3 (the new function). The feasible organizations between the Mino and Maxo represent various combinations of the Options of Fig. 19 for the three planning areas. Example Petri Nets of the Mino and Maxo are shown in Fig. 24.

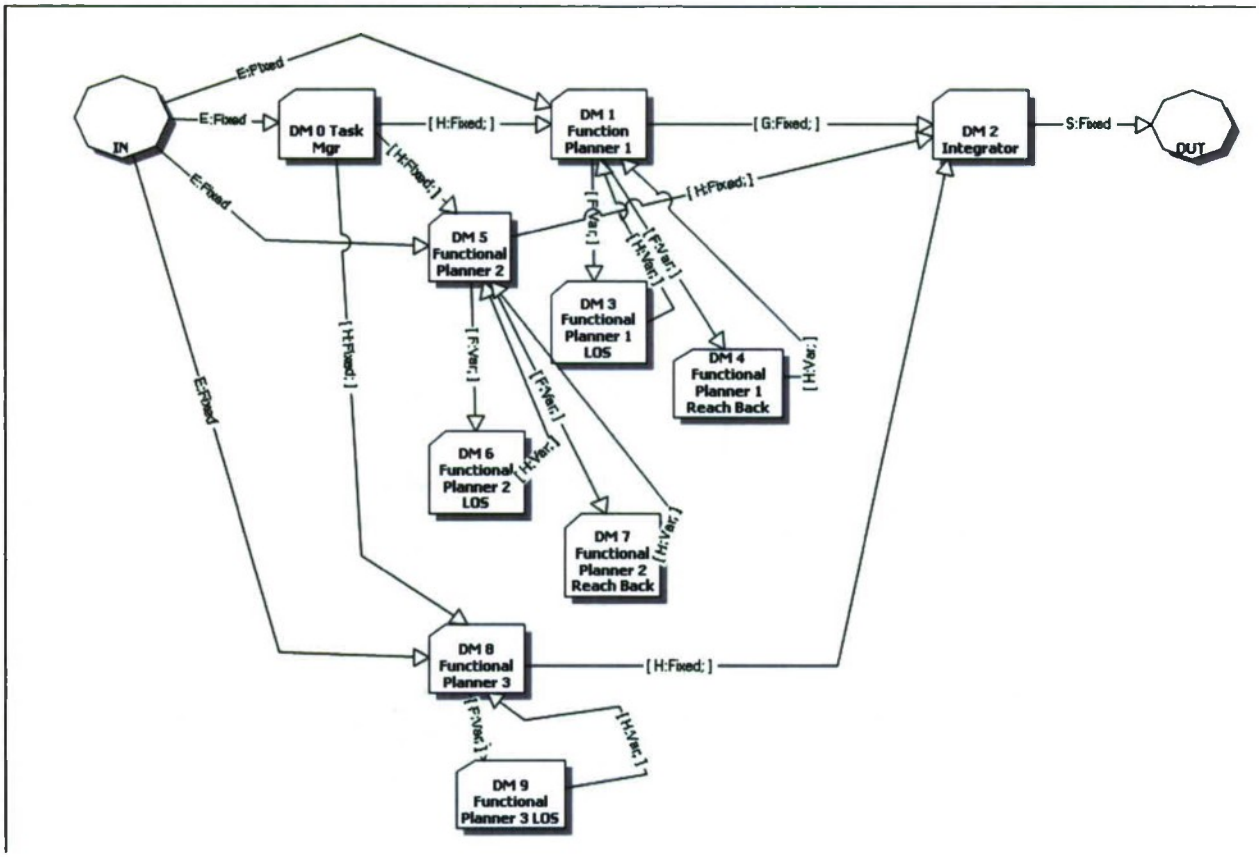


Fig. 22. CAESAR III Design View of MOC for Analysis of Alternatives

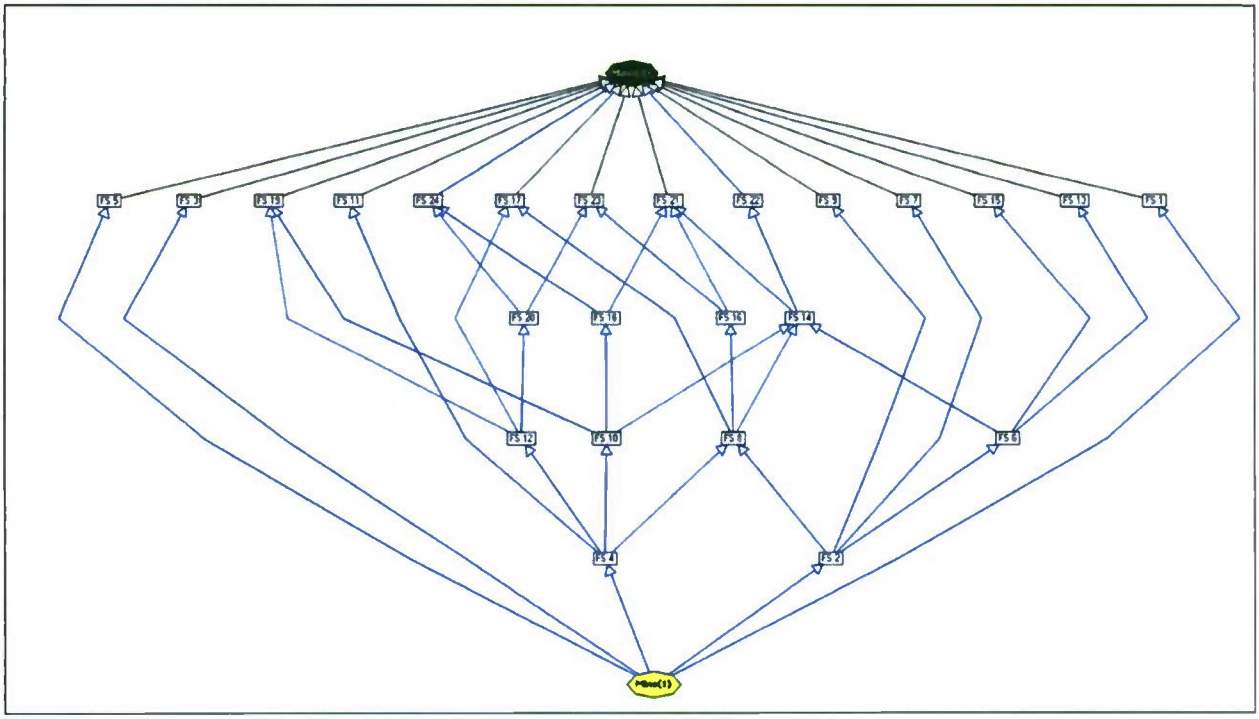


Fig. 23. Lattice from Design of Fig. 18

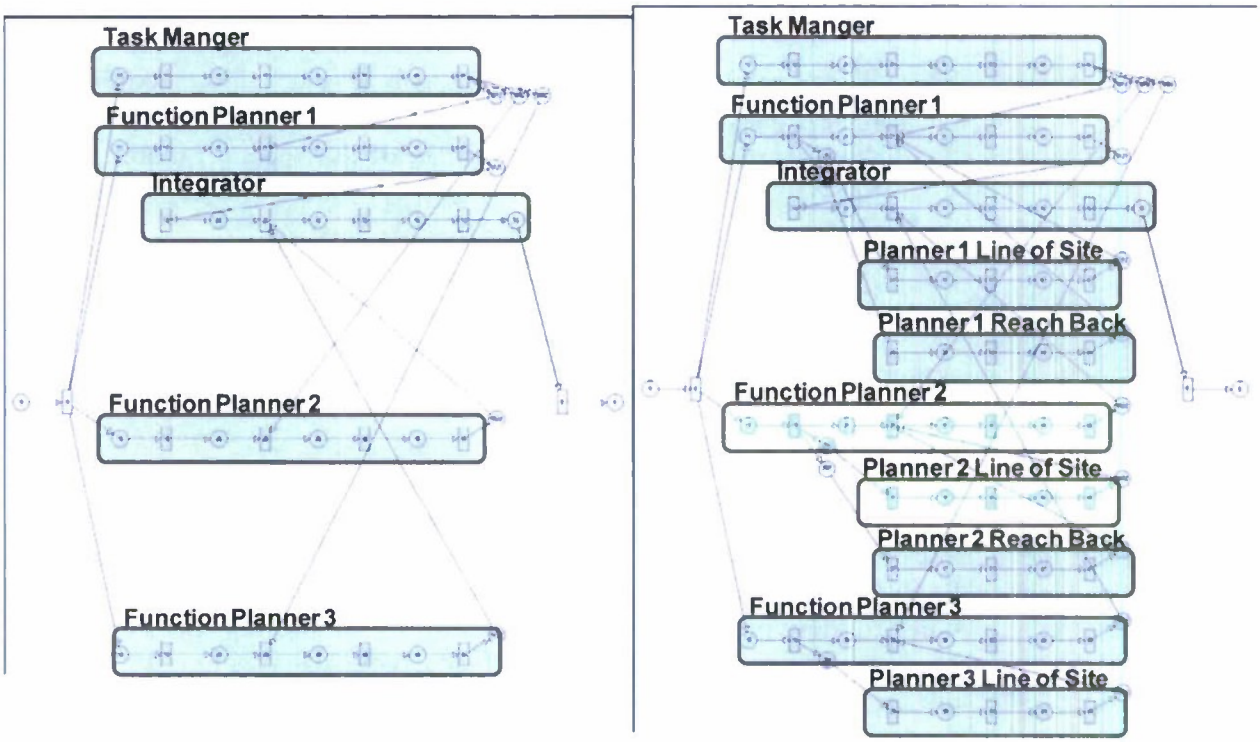


Fig. 24. MINO and MAXO Petri Nets

The derivation of all the possible organization structures is not sufficient to complete a design of the future MOC. The constraints of the maximum number of people that can be accommodated on the Command ship and the Line of Sight vessels along with their level of expertise act as constraints on the design. Acceptable solutions will be based on availability of the needed expertise at the various locations, the availability of the required communications between various locations and the acceptability of cyber risk. In addition, the Commander the MOC is supporting will have preferences as to the key analysis and planning expertise he will require, particularly on the Command Ship. These constraints and preferences result in a possible workflow that could be used to design a MOC organization, as follows.

1. Identify "highest" command mission MOC anticipated to support
2. Determine the external interfaces (information flow requirements) including the Superior Command, Subordinate Commands, Peer Commands (Coordination and Synchronization), and External Agencies.
3. Determine the key analysis and planning expertise commander will require (Commander selection drives staff requirements).
4. Determine key analyst and planner support staff requirements.
5. Use CAESAR III to develop alternative organizations based on organic staff, line-of-site staff, and reach back staff.
6. Calculate the number of personnel required for all organic staff (staff on Command Ship).
7. Shift functions from organic to Line of Sight until within the maximum number personnel.

8. Calculate number of personnel now required Line of Sight.
9. Select functions for reach back support until number of personnel are below maximum for the Line of Sight location. There should now be one or more feasible organization designs based on the desires of the Commander and the constraints of the space available on the Command Ship and the Line of Sight locations.
10. Use the selected CAESAR III design to expand the task graph and create Operational and System Viewpoint descriptions of the configuration and examine the potential vulnerabilities from cyber effects.
11. Make the final selection of the architecture design by considering the Cyber Vulnerability risks.

3.4 Summary

In this research task GMU developed a fundamental concept of augmentation that can be thought of as a micro view of scalability. It defined Augmentation by Staff and Augmentation by Function and identified some of the key issues, particularly with respect to coordination that these concepts entail. A macro view of the MOC design problem was develop that focused on the different levels of command organization that apply to the types of situations in which a MOC may be used. This analysis lead to the understanding that there are key constraints that must be considered when determining how to change the scale of a MOC including the amount of coordination induced workload, the limited amount of space on the Command Ship and other vessels, the location of required expertise, and the type of communications infrastructure required for a particular MOC configuration. Then GMU focused on the use of tools to create MOC designs. The CAESAR III tool was used to examine alternative MOC organizational designs including focusing on using the micro and macro concepts. Finally a MOC architecture design process was developed and illustrated that is based on the DOD Architecture Framework (DODAF) for describing not only the MOC organization, but the services, systems, and networks that support those organizations. This description supports trade-off analysis between location of the staff of a MOC and potential behavior, performance, and risk of cyber effects in contested environments.

4. VERIFICATION OF INCONSISTENCY AND REDUNDANCY IN MARITIME LAWS

4.1 Introduction

Different sets of law apply to a ship, or a fleet of ships, when it is operating at sea, e.g., Admiralty Law, Law of the Sea, etc. These laws come from different sources (e.g., international organizations, nation states, etc.) and their applicability changes from one geographical location to the other and also with the time of their application. A Maritime Lawyer, on board the ship or a ship in the fleet, is required to look into these law books to identify the applicability of action(s) in a given situation.

At present, this process is done manually by the Maritime Lawyers, who search for all the rules that are relevant to answer queries about applicability of action(s), considering the geographical boundaries, temporal aspects, and other attributes of the sitauation. From the set of identified rules, they figure out the applicability of action(s) and present the results to the commander of the ship or fleet.

The entire process of information foraging and deciding the course of action from the applicable rules is a time-consuming activity. It becomes even more difficult for some time-critical decisions that need to be made, especially in a rapidly changing situation. Also, there is no guarantee that the rules are complete and consistent across all the law books, resulting in an additional task on the Maritime Lawyers to verify the completeness and consistency of the rules before arriving at a conclusion. Consequently, a computer-aided approach is needed that facilitates the process of information foraging and checking for possible conflicting, incomplete, and redundant cases across disparate law books.

We present a new technique for the identification of inconsistency and redundancy within rules selected from different Maritime Law sources. As part of this effort, a knowledge management system was developed using Zotero [1] – a Firefox plug-in for knowledge organization. This OPLAW management system provides storage facility for various Maritime Laws and rules within them and facilitates selection of rules specific to a particular task, serving as a source of input to our developed verification technique. The technique extends an earlier approach proposed by Zaidi and Levis [2] by removing some of the limitations of their methodology. A discussion on some weaknesses in the previous approach, namely in the algorithms for identification of inconsistent and redundant rules, is presented in the next section. The proposed verification technique makes use of Production Systems as the formal knowledge representation scheme for Maritime Laws, where pre- and post-conditions of a production rule are given as First-Order expressions. A Maritime Law can be regarded as a Condition-Action rule which helps a Maritime Lawyer to determine when, where, and how certain actions could be taken. It is because of this fact that Production Systems are used as a knowledge representation scheme; the Production System representation is useful in action selection problems.

4.2 Related Work

There have been a number of tools and techniques for validation and verification (V&V) of rule-based systems that employ production rules as the representation scheme. A large number of these proposed approaches make use of Petri Nets for the V&V task because they provide exactly the same operational formalism as Production Systems. Zisman [3] was the first to note this similarity and use Petri Nets for the verification task. Some of the earlier techniques that utilize Petri Nets were proposed by Giordana and Saitta [4]; Zhang and Nguyen [5], [6]; Liu and Dillon [7] and Agarwal and Tanniru [8]. Zaidi and Levis [2] discuss the weaknesses of these earlier approaches. The technique proposed by them divides the set of verifiable properties into two classes: structural and behavioral. The behavioral set of properties includes redundancy and inconsistency. Their approach requires construction of a state space graph (also known as reachability graph or occurrence graph) for the Petri Net representation of a rule-base, and a search within it for identification of these properties. The methodology is intended for the case where the initial marking of the Petri Net is unknown. A heuristic is used to overcome the combinatorial enumeration problem that results from the unknown input assumption for generation of the state space graph and its analysis. However, the approach proposed for behavioral analysis is, in general, too complex to be used in practice. A number of other techniques were proposed afterwards, but most of them have certain problems of their own, which are presented in the following discussion.

Yang et al. [9] propose an approach that uses the incidence matrix of the Petri Net representing a rule-base to verify properties within it. The approach is based on an ordering of rules according to some criteria defined by the authors before representing them with a Petri Net and using its corresponding incidence matrix for the verification task. Furthermore, the technique proposed by them works with a restricted rule format. Their approach does not have the capability for verification of properties in chains of rules.

He et al. [10] and Yang et al. [11] propose an approach that uses ω-net (a special type of low-level Petri Net) for rule-base representation and the corresponding reachability graph of the ω-net for verification of properties. The issue in both the proposed techniques is that the structure of the reachability graph for a ω-net ignores paths to reachable markings, resulting in additional work for the accurate detection of those properties [12].

Wu et al. [13] presented the use of Colored Petri Nets for rule-base representation and its reachability graph analysis for the verification task.

Ding et al. [12] recently proposed a technique that extends the approach of He et al. [10] by generating a backward reasoning forest of reachable markings in reachability graphs. Their purpose for doing this is to explicitly present and record the missing reachable path information within the reachability graph of a ω-net. However, their approach has the limitation of working with only those rule-bases where rules are expressed as Horn clauses.

In the sequel, we present the new methodology that overcomes the weaknesses of previous approaches. The approach extends the work proposed by Zaidi and Levis [2] by presenting a formal technique of model checking to capture redundant and inconsistent cases in a rule set across several rule sets.

4.3 Verifiable Characteristics

The Production System representing the set of rules from across different Maritime Laws is likely to have inconsistency, redundancy, etc., since the constituent rules come from different law sources. A rule from one Maritime Law may allow an action, whereas another rule addressing the same situation from a different Maritime Law may prohibit it. There can also be cases in which two Maritime Laws, applicable in a given situation, may say the same thing. Under these circumstances, a Maritime Lawyer has to suggest if an action can be taken or not. In other words, he/she has to be certain about his/her decisions after careful consideration of the various characteristics that may appear in the set of applicable Maritime Laws. This section outlines such characteristics and provides a formal description of them by demonstrating them using Production Systems as the representation technique for rules. It should be noted that the formal description of the set of properties, presented in this section, has been compiled from different V&V approaches that were reported in the literature. The discussion that follows makes use of a hypothetical set of production rules for describing these properties. Examples of such properties in Maritime Laws are provided in the following section 4.5.

4.4 On Rules

Inconsistent (Conflicting) Rules

Inconsistent rules result in conflicting facts [10] derived from them. The execution of a rule set having inconsistent rules results in new facts that either conflict with one another or conflict with existing fact(s). In other words, it can be said that a rule set free of inconsistent rules never infers conflicting information.

Example 1:
Consider the two rules shown below. Given A, B, C, and D (as facts), these two rules become applicable, leading to two conflicting conclusions (P and ¬P) if executed.

$$\begin{aligned} \text{Given facts:} \quad & A \wedge B \wedge C \wedge D \\ \text{Rule 1:} \quad & A \wedge B \rightarrow P \\ \text{Rule 2:} \quad & C \wedge D \rightarrow \neg P \end{aligned}$$

Example 2:
Consider another set of facts and rules presented below. Based on the given information, the first rule becomes applicable leading to a conclusion which makes the second rule applicable as well; that leads to a conclusion (¬A) contradicting the original information (A).

$$\begin{aligned} \text{Given facts:} \quad & A \wedge B \wedge D \\ \text{Rule 1:} \quad & A \wedge B \rightarrow P \\ \text{Rule 2:} \quad & P \wedge D \rightarrow \neg A \end{aligned}$$

Redundant Rules

Redundancy refers to the presence of multiple copies of the same rule or the presence of set of rules that lead to the same effect (output), from the same input conditions [2].

Example 1:
Consider the set of rules shown below. The two provided rules are identical except that their input conditions are arranged differently. Based on the given set of facts (A and B), which makes their input conditions true, the same conclusion is derived by their execution.

Given facts: $A \wedge B$
Rule 1: $A \wedge B \rightarrow P$
Rule 2: $B \wedge A \rightarrow P$

Example 2:
The rule set shown below is another example of redundancy in which the first three rules have the same effect as the last rule, given the same input conditions (A, B, C, and D).

Given facts: $A \wedge B \wedge C \wedge D$
Rule 1: $A \wedge B \rightarrow P$
Rule 2: $C \wedge D \rightarrow Q$
Rule 3: $P \wedge Q \rightarrow R$
Rule 4: $A \wedge B \wedge C \wedge D \rightarrow R$

Incompleteness

Incompleteness refers to a situation when there is not enough information in the rule base to answer a query of interest [14]. Some of the previous approaches in V&V literature have further categorized this case into Rules with Unknown Conditions and Rules with Useless Conclusions [2], [6].

Rules with Unknown Conditions

This incompleteness refers to a situation when the system does not have rules that can reach some desired conclusion (i.e. answer a query).

Example:
Consider the set of rules shown below. Based on the given set of facts (A, B, C, and D), it is not possible for the system to make an assertion R, since one of the input conditions (S) of the last rule, which can assert R, is unknown. This situation may have originated because of some missing rule that can make assertion S from the input information or due to insufficient input information.

Given facts: $A \wedge B \wedge C \wedge D$
Rule 1: $A \wedge B \rightarrow P$
Rule 2: $C \wedge D \rightarrow Q$
Rule 3: $P \wedge Q \wedge S \rightarrow R$

Useless Rules

They refer to the rules that do not produce useful results. In other words, these are the rules whose output is not of interest to the user and there is no rule or set of rules leading from their outputs to useful conclusions [2].

Example:

Consider a case, where the set of facts and rules are as provided below and a user is interested to know if assertion P can be made or not (i.e., ?P). From the given information, it can be seen that the system can reach to a decision about P using the first rule, however, the remaining rules are applicable as well, whose outputs are not relevant to the query in this case.

Given facts: $A \wedge B \wedge C \wedge D$
Rule 1: $A \wedge B \rightarrow P$
Rule 2: $C \wedge D \rightarrow Q$
Rule 3: $A \wedge B \wedge C \wedge D \rightarrow R$

Goal: ?P

Circular Rules

Circularity refers to the presence of rules that will lead a system to get trapped in executing them again and again [2].

Example 1:
Consider the two rules shown below. Given A (as fact), the first rule becomes applicable leading to a conclusion which makes the second rule applicable as well. The conclusion of the second rule leads back to the first rule to become applicable again.

Given fact: A
Rule 1: $A \rightarrow B$
Rule 2: $B \rightarrow A$

Another case of circularity is defined as the presence of rules that are circularly dependent on each other resulting in a deadlock [2].

Example 2:
The set of rules shown below illustrates the case of a deadlock. Given A and B as facts, the first rule cannot execute since it requires an input condition R that can be obtained by executing the second rule, which in turn requires the execution of the first rule to make an assertion P to become applicable in the given situation.

Given facts: $A \wedge B$
Rule 1: $A \wedge R \rightarrow P$
Rule 2: $B \wedge P \rightarrow R$

Subsumed Rules

Subsumption is a special case of redundancy in which one rule (or a rule set) is less restricted than the other. In other words, the antecedent conditions of one rule (or a rule set) are a subset of the antecedent conditions of the other, both leading to the same conclusion [2].

Example:
Consider the set of facts and rules shown below. The first rule subsumes the other, since it takes a portion of the input conditions of the second rule to reach the same conclusion on execution.

$$\text{Given facts: } A \wedge B$$
$$\text{Rule 1: } A \rightarrow R$$
$$\text{Rule 2: } A \wedge B \rightarrow R$$

4.5 Sample Maritime Laws

In this section, we present a small set of rules from different Maritime Law sources to illustrate the properties that have been described in the previous section within these rules. The rules and test queries were taken from a maritime scenario report, titled "Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0", which was created and provided by Pacific Science & Engineering Group, Inc. (http://www.pacific-science.com). The formal representation of the rule set presented in this section was, however, not a part of the report and was created manually for the verification technique. The rules, from across different Maritime Laws, are shown as a single set in the following discussion. For the sake of brevity, in the following presentation, we do not mention their law sources, article numbers, and other relevant information provided in the scenario report. We will refer to these rules by the numbers assigned to them in Table 1.

Formal Representation

The proposed verification technique makes use of Production Systems as the formal representation of a rule set, where pre- and post-conditions of a production rule are given as First-Order expressions. The reason for choosing First-Order Logic (FOL) to encode the conditions and actions of production rules is because of its adequacy of the representation, i.e., its expressive power [15]. Since a FOL formula having a finite set of constant symbols can be converted to an equivalent Proposition Logic (PL) formula – a process known as *propositionalization* [16] – we transform the FOL statements in production rules to PL statements by instantiating their variables with known constant values. Table 2 shows the set of *propositionalized* production rules corresponding to the sample rules presented in Table 1. The strings that appear outside the parentheses are the predicates, which together with the constant values (in the parentheses) make up a propositional statement. We have used a shorthand representation to represent the long strings of constant values. For example, V1 in the logical representation only represents some vessel instance that is in sight or observation. Similarly, C1 is the shorthand representation for a country instance in the region where a rule is applied. It should also be noted that a proposition and its negation are represented by two separate propositional statements in our approach as opposed to the representation which makes use of the symbol '¬' to represent a negation. The action propositions of Rules 6(a) and 5(b) are an

example of such a case, which are negation of one another and are represented by two separate propositional statements (i.e., pursue(V1, C1) and not_pursue(V1, C1)).

Table 1: Sample Rules (from Scenario Report)

1. No biometric data is to be gathered without prior oral consent from individuals of interest.
2. Only ships that have been declared pirate ships by the policing entity are allowed to be searched.
3. Labeling a ship as non-compliant does not authorize U.S. forces to search vessel.
4. It is prohibited to follow any suspected pirate ship into territorial waters unless: (a) Owner of territorial waters has specifically requested U.S. assistance, and has given permission for U.S. vessel to enter their water, or (b) Pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.
5. (a) A ship shall be considered a pirate vessel if and only if hostile acts or hostile intents against U.S. forces are directly observed by U.S. forces. (b) Under no circumstances is it authorized to pursue a pirate vessel into territorial waters.
6. It is not permitted to pursue a pirate ship into territorial waters unless: (a) Express consent has been given by the coastal state prior to entry into their territorial waters, or (b) The hostile act was committed against U.S. forces or U.S. property, or (c) The hostile act was committed in U.S. territorial waters.
7. U.S. forces are authorized to search any vessel that has been declared a pirate vessel by a U.S. or foreign entity.
8. (a) The collection of biometric data is authorized for ships declared to be pirate ships or defined as non-compliant. (b) The collection of biometric data is authorized for all other ships, per the consent of the individuals of interest.

Table 2: Formal Representation of Sample Rules (of Table 1)

1. normal_vessel(V1) $\wedge$ biometric_data(D1) $\wedge$ permitted(P1) $\rightarrow$ collect(D1)
2. pirate_vessel(V1) $\rightarrow$ search(V1)
3. non_compliant_vessel(V1) $\rightarrow$ not_search(V1)
4. (a) pirate_vessel(V1) $\wedge$ territorial_water(C1) $\wedge$ position(V1, C1) $\wedge$ entry_request(C1) $\wedge$ permitted(C1) $\rightarrow$ pursue(V1, C1) (b) pirate_vessel(V1) $\wedge$ territorial_water(C1) $\wedge$ position(V1, C1) $\wedge$ hostile_act(V1) $\rightarrow$ pursue(V1, C1)
5. (a) normal_vessel(V1) $\wedge$ hostile_act(V1) $\rightarrow$ pirate_vessel(V1) pirate_vessel(V1) $\rightarrow$ normal_vessel(V1) $\wedge$ hostile_act(V1) (b) pirate_vessel(V1) $\wedge$ territorial_water(C1) $\wedge$ position(V1, C1) $\rightarrow$ not_pursue(V1, C1)
6. (a) pirate_vessel(V1) $\wedge$ territorial_water(C1) $\wedge$ position(V1, C1) $\wedge$ entry_request(C1) $\wedge$ permitted(C1) $\rightarrow$ pursue(V1, C1) (b) pirate_vessel(V1) $\wedge$ territorial_water(C1) $\wedge$ position(V1, C1) $\wedge$ hostile_act(V1) $\rightarrow$ pursue(V1, C1) (c) pirate_vessel(V1) $\wedge$ territorial_water(US) $\wedge$ position(V1, US) $\wedge$ hostile_act(V1) $\rightarrow$ pursue(V1, US)
7. pirate_vessel(V1) $\rightarrow$ search(V1)
8. (a) pirate_vessel(V1) $\wedge$ biometric_data(D1) $\rightarrow$ collect(D1) non_compliant_vessel(V1) $\wedge$ biometric_data(D1) $\rightarrow$ collect(D1) (b) normal_vessel(V1) $\wedge$ biometric_data(D1) $\wedge$ permitted(P1) $\rightarrow$ collect(D1)

Shorthand Representation of Formal Rules

These long propositional statements of machine-readable rules can be represented by a shorthand representation using symbols. The symbols and their corresponding propositions are listed in Table 3. These symbols will be used in the remaining discussion instead of the corresponding propositions they represent.

Table 3: Set of Propositions Used in the Formal Representation of Maritime Laws

Symbols	Propositions
P1	normal_vessel(V1)
P2	biometric_data(D1)
P3	permitted(P1)
P4	collect(D1)
P5	pirate_vessel(V1)
P6	search(V1)
P7	not_search(V1)
P8	non_compliant_vessel(V1)
P9	territorial_water(C1)
P10	position(V1, C1)
P11	entry_request(C1)
P12	permitted(C1)
P13	pursue(V1, C1)
P14	hostile_act(V1)
P15	not_pursue(V1, C1)
P16	territorial_water(US)
P17	position(V1, US)
P18	pursue(V1, US)

Examples of Various Cases within Maritime Laws

Example 1:
This example uses a test query (shown below), to demonstrate a case in which conflicting conclusions are reached from two different applicable Maritime Laws.
"Is it allowed to follow a pirate vessel in territorial waters, if it commits a hostile act?"
Formally,

Given Facts: $P5 \wedge P9 \wedge P10 \wedge P14$
Goal: ?P13, ?P15

The rules relevant to answer this query are Rule 5(b) and Rule 6(b). However, they totally contradict one another since one rule prohibits following pirates in territorial waters (P15) in any case, whereas the other allows such an action (P13) in the given condition, i.e., when the vessel commits a hostile act against U.S. forces.

$(P5 \wedge P9 \wedge P10 \wedge P14)$
$\underline{P5 \wedge P9 \wedge P10 \rightarrow P15}$ (Rule 5b)
$P15$

$$\frac{(P5 \wedge P9 \wedge P10 \wedge P14) \quad P5 \wedge P9 \wedge P10 \wedge P14 \rightarrow P13}{P13} \qquad \text{(Rule 6b)}$$

Example 2:
This example presents an instance of redundancy, i.e., a case where rules from different Maritime Laws say the same thing in a given situation. It uses the following query to illustrate the said case.
"Is it permitted to stop and search a pirate vessel?"
Formally,

Given fact: P5
Goal: ?P6

The rules relevant to this query are Rule 2 and Rule 7, which are identical (in terms of both input condition and conclusion). The two rules conclude that U.S. forces are allowed to search any vessel which has been declared pirate.

$$\frac{(P5) \quad P5 \rightarrow P6}{P6} \qquad \text{(Rule 2)}$$

$$\frac{(P5) \quad P5 \rightarrow P6}{P6} \qquad \text{(Rule 7)}$$

4.6 Technical Approach

In this section, we describe the process of transforming production rules to a Petri Net model and present a methodology for its verification. An overview of our proposed approach is shown in Fig. 25. The formally represented set of rules is first transformed into an equivalent Petri Net representation, which is then initialized based on the given input situation. The structural and behavioral analysis of the Petri Net is then performed to look for properties of the net corresponding to useless, incomplete, cyclic, redundant, and/or inconsistent cases. The structural analysis is done by the approach proposed in [2] to capture useless, incomplete, and cyclic cases. The behavioral analysis is done by the new technique, presented in this section, which employs a formal model checking methodology for the verification of inconsistent and redundant cases. The part of the approach, which was the main focus of this research, is highlighted by colored arrows in the figure.

Since our technique extends an earlier approach proposed in [2], we make use of the same Petri Net representation described in it. The material presented on the Petri Net representation has been taken from [2] and on formal model checking theory from [17], [18] with minor editorial changes.

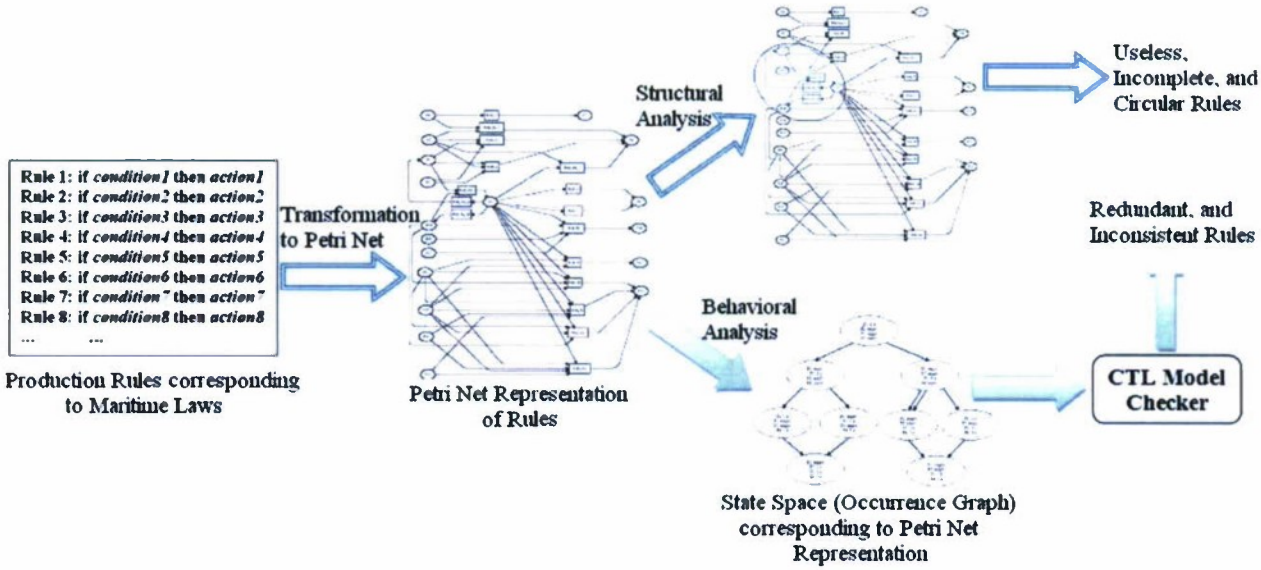


Fig. 25. Overview of Verification Approach

Petri Net Representation of Rules

Our approach works with *propositionalized* production rules, whose antecedents are organized in Disjunctive Normal Form (DNF) and the conclusions in Conjunctive Normal Form (CNF). This assumption does not impose any restriction, since any schema constructed with the help of logical connectives and operators, can be converted to an equivalent Disjunctive Normal Form or an equivalent Conjunctive Normal Form. An example of such a rule is:

$$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n \rightarrow \beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_m$$

where α and β are conjunctions and disjunctions of atomic propositions, respectively. The rule specified above can further be split into the following set of rules:

$$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n \rightarrow \beta_1$$
$$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n \rightarrow \beta_2$$
$$\ldots$$
$$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n \rightarrow \beta_m$$

It may also be split into the following set of rules:

$$\alpha_1 \rightarrow \beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_m$$
$$\alpha_2 \rightarrow \beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_m$$
$$\ldots$$
$$\alpha_n \rightarrow \beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_m$$

It follows that the rule expressed in form (5.1) can be split into the following (n * m) rules:

$$\alpha_1 \rightarrow \beta_1$$
$$\alpha_1 \rightarrow \beta_2$$
$$\ldots$$
$$\alpha_1 \rightarrow \beta_m$$
$$\alpha_2 \rightarrow \beta_1$$

$$\alpha_2 \rightarrow \beta_2$$

$$\ldots$$

$$\alpha_n \rightarrow \beta_m$$

This provides us with a set of rules in which the antecedents are conjunction of one or more propositions and the consequents are disjunction of one or more action propositions. The normalized set of rules is then transformed into an equivalent Petri Net representation. For each rule, a Petri Net is generated where the conditions of a rule are represented by input places, satisfying the definition of conjunctive operators, and the conclusions by output places. Each output β_i, with k individual action propositions (connected by disjunction), is required to be represented by additional (2^k-1) output transitions leading to k places (representing those individual propositions), that corresponds to all possible combinations of action propositions according to the definition of disjunctive connectives. Figure 26 shows a representation for such a case. It should be noted that the Petri Net model shown in Fig. 26 corresponds to a hypothetical rule, which is not a part of the sample rule set and is presented here to explain the concept.

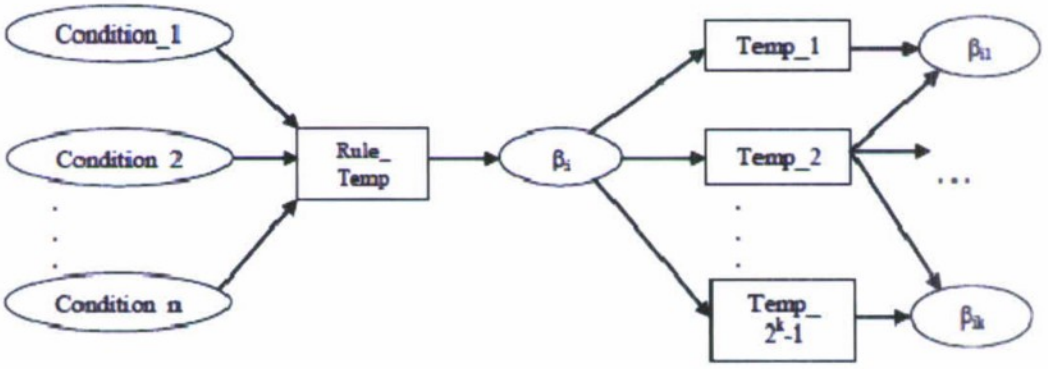


Fig. 26. Petri Net Representation for Disjunction in Conclusion

The labels on transitions correspond to the rules they represent. The negation of a proposition (i.e., ¬P) is represented by a place which is different from the place representing positive proposition (i.e., P). A token in a place represents the truth assignment of a proposition. If all the input places of a transition, representing the input conditions of a rule, have a token then the transition (rule) is enabled and can fire (execute) making the output place (consequent) true. Finally, a single Petri Net representation for the entire rule set is obtained by merging all the common places of Petri Nets representing individual rules. An example of such a process using Rule 2 and Rule 7 (above) is shown in Fig. 27.

The Petri Net representation for the sample Maritime Laws was generated based on the defined mapping. The resulting Petri Net is shown in Fig. 28. It will be used for structural and behavioral analysis to explore various properties. For brevity, we refrain from presenting details on the structural analysis approach. Interested readers can refer to [2] for information on it; for convenience, it is included in this report as Appendix A.

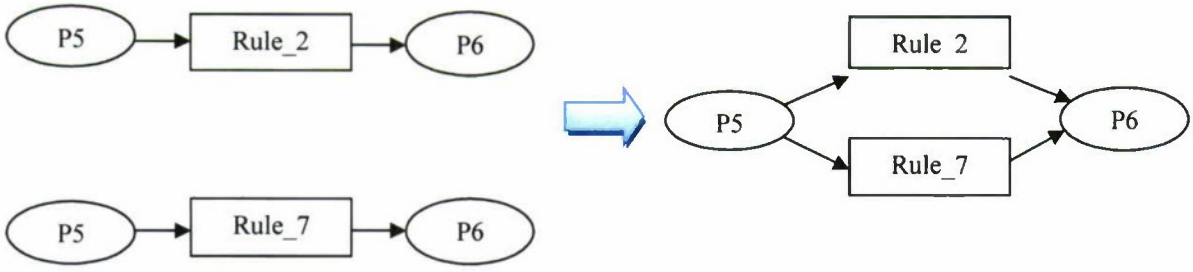


Fig. 27. Petri Net Before and After Merging Common Places

Behavioral Analysis of Petri Net

Behavioral analysis, also known as dynamic analysis, makes use of an Occurrence Graph that corresponds to the dynamics (or execution) of the Petri Net, to capture properties corresponding to redundant and inconsistent cases. It uses the state space of a Petri Net system to look for patterns among states or inside a state description that correspond to the two cases. Before we discuss such patterns, let us first define some of the terms that are used extensively in the remaining discussion.

Definition (Marking):- A marking M of Petri Net model is a mapping that assigns a non-negative number to each place P in the net [19].

$$M: P \rightarrow \mathbb{N} \text{ where } \mathbb{N} = \{0, 1, 2, \ldots\}$$

The marking of a Petri Net can be expressed as a vector with a specified ordering of the places.

Definition (Occurrence Graph):- An Occurrence Graph (OG), also known as state space, for a Petri Net with an initial marking M_0 (also called a Petri Net System) is a directed-graph that consists of:
– a set of nodes corresponding to the set of reachable markings from M_0,
– a set of arcs between nodes that correspond to transitions from one marking to another [2]. The labels on arcs represent the corresponding transitions of the Petri Net.

An example of an Occurrence Graph is shown in Fig. 29. The example OG corresponds to the Petri Net of Fig. 28 with initial marking M_0 = [1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0], where the places in M_0 are arranged in ascending order. The given initial marking M_0 corresponds to a hypothetical input situation in which a vessel is encountered whose crew agrees to give their biometric data, i.e.,

$$\text{Given facts: } P1 \wedge P2 \wedge P3$$

The set of places that are marked with a strictly positive number in the new marking M_1 of the example OG correspond to the propositions that evaluate as true in the given situation. The labels on the arcs represent the rules that lead from the initial marking M_0 to the new marking M_1.

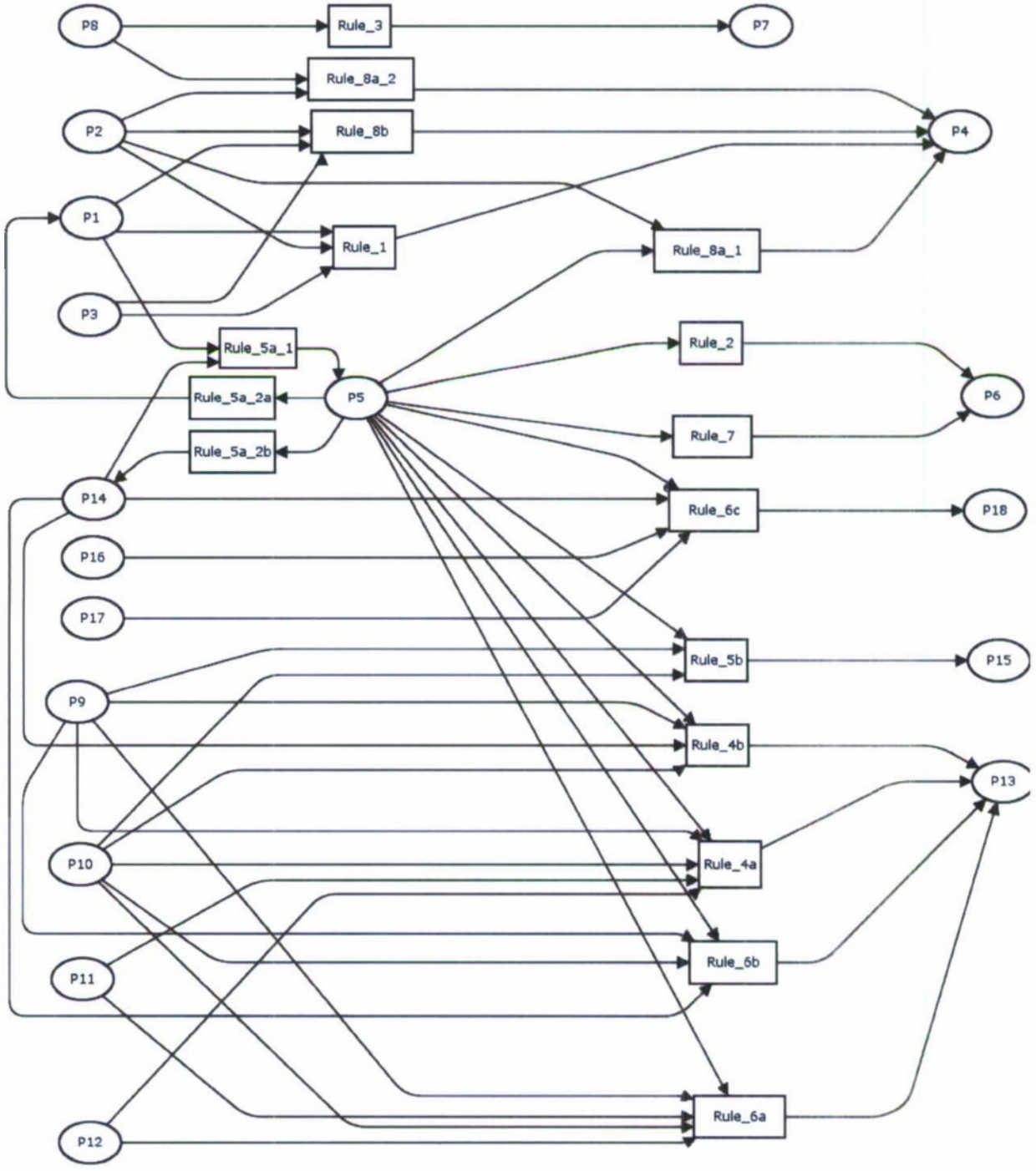


Fig. 28. Petri Net Representation of Maritime Laws

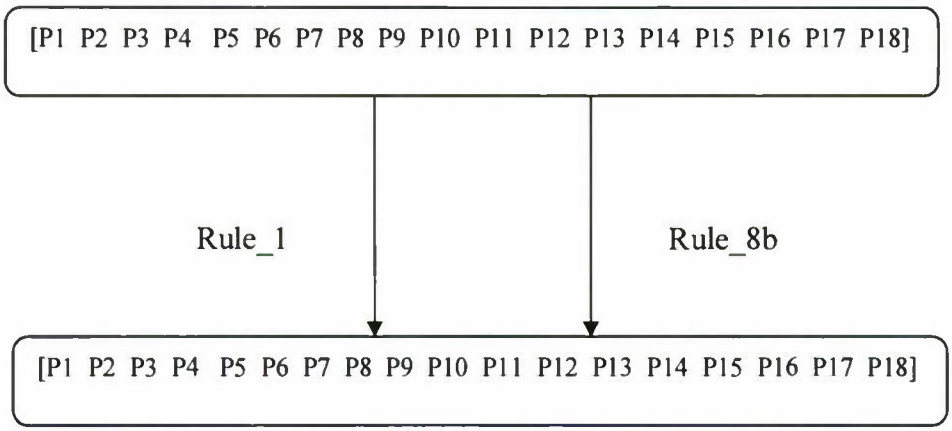


Fig. 29. Occurrence Graph for Petri Net (of Fig. 28) with M_0 = [1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

Patterns of Inconsistent and Redundant Cases in Occurrence Graphs

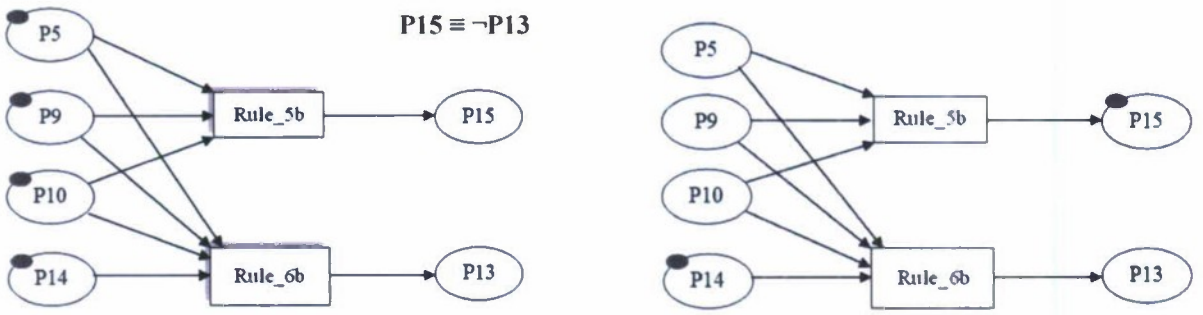
In this section, we present the patterns (or composition) of redundant and inconsistent cases within Occurrence Graphs. Such patterns can help in formulating the two properties as formal logical expressions, which can be checked against the state space of a Petri Net representing Maritime Laws.

(a) Inconsistent Rules

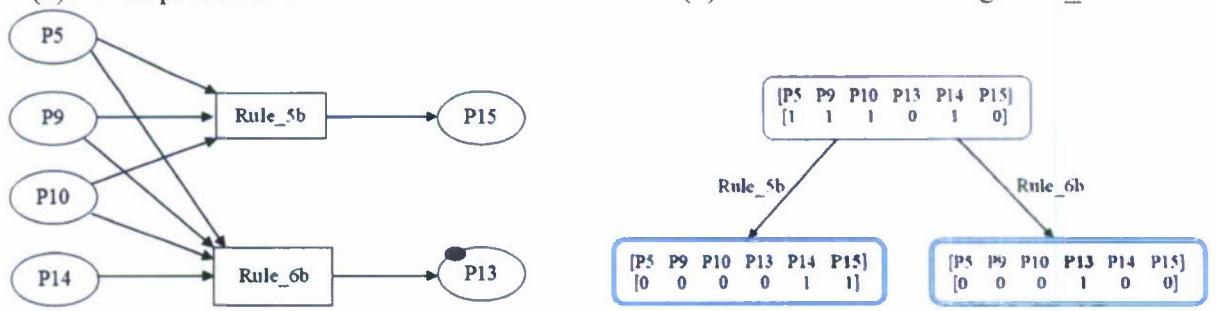
As per the definition of inconsistency, such a case occurs when two conflicting concepts become true in a given situation. In a Petri Net, these cases appear in the form of two conflicting places getting marked by a token in the set of firing (or execution) sequences from M_0 (where M_0 represents the initial marking corresponding to a given situation). As a result, the two conflicting places will get marked in the corresponding OG as well. An example of such a case appearing in a Petri Net and the corresponding OG is shown in Fig. 30(a-d). The Petri Net model shown in Fig. 30(a-c) corresponds to Rules 5(b) and 6(b). Assume that the input conditions of both these rules are true, which makes both of them applicable. The two execution sequences, corresponding to the enabled transitions, result in conflicting places (P13 and P15) getting marked, implying a contradiction in conclusion. Figure 30(b-c) shows such a case by the presence of tokens in P13 and P15 for the two execution sequences. This conflicting behavior is represented by the corresponding OG as well (refer to Fig. 30d), which shows contradiction in the form of P13 and P15 getting marked in two different reachable states (or markings)

(b) Redundant Rules

In a Petri Net, redundancy appears in the form of several paths from place(s) with multiple outputs to place(s) with multiple inputs [2]. In an Occurrence Graph of a Petri Net model, such cases appear in the form of multiple distinct paths from one state (marking) to another state [2]. An example of a Petri Net and the OG representation for a redundant case is given in Fig. 31(a-d). The Petri Net model shown in Fig. 31(a-c) corresponds to Rules 4(b) and 6(b). Assume a situation in which the input conditions of these rules are true, in which case both the rules will lead to same conclusion on execution. The transition firing process for the two rules is shown in Fig. 31(b-c). An Occurrence Graph representation corresponding to the Petri Net is provided in Fig. 31(d), which shows redundancy in the form of two distinct paths from one state to another.

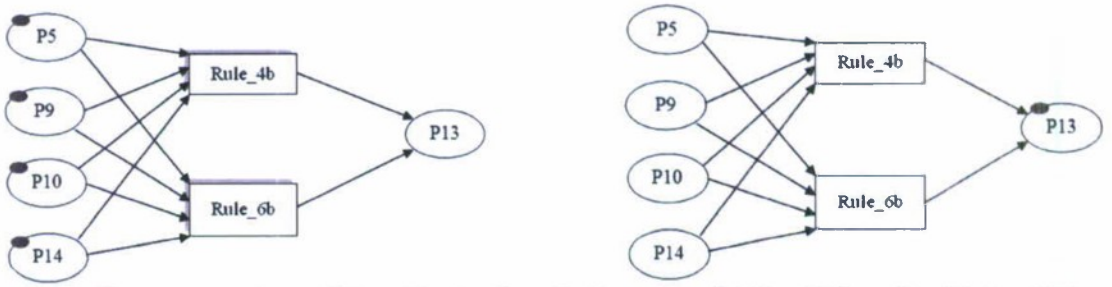


(a) PN Representation of Two Inconsistent Rules (b) Petri Net after Firing Rule_5b

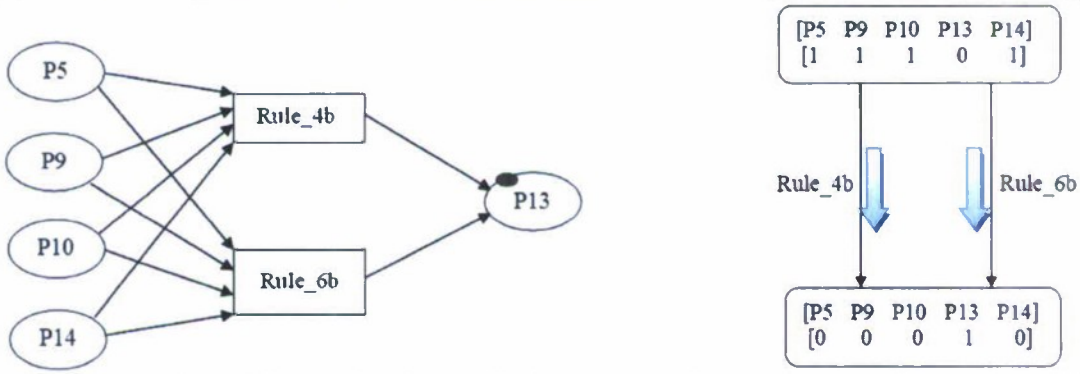


(c) PN after Firing Rule_6b (d) Occurrence Graph corresponding to PN of (a)

Fig. 30. Pattern of Inconsistent Rules



(a) Petri Net Representation of Two Redundant Rules (b) Petri Net after Firing Rule_4b



(c) Petri Net after Firing Rule_6b (d) Occurrence Graph corresponding to Petri Net of (a)

Fig. 31. Pattern of Redundant Rules

4.7 Model Checking

Model Checking is an approach for formal verification of finite state concurrent systems. It starts by specifying a property of a system as a formal logical expression and then interprets this logical expression over the state space of the system to establish if the given property holds in the system's behavior or not. In other words, it establishes, using an exhaustive search of the state space, if the system's behavior is a model of the input logical expression or not. The process always ends with a yes or no answer indicating the presence or absence of the specified property within the system. The process of model checking requires the following steps:

a) *Modeling:* The first step is to represent the system by a model accepted by the model checker.

b) *Specification:* The second step requires specification of properties that need to be checked in the system. The properties are normally expressed by temporal logic, which can assert how the behavior of a system changes over time, or as a result of chains of rules being executed in a rule set.

c) *Verification:* The verification step is automated. It exhaustively searches the state space of modeled system to check if the specified property exists in the system. It provides an error trace in case of a property violation that can be used to identify the point of error.

ASK-CTL is an extension of Computation Tree Logic (CTL), a class of temporal logics, which is used to model check Petri Nets. It is used for specification of properties that need to be checked in systems represented by Petri Net models. The specified properties are checked against the state space (or Occurrence Graph) of a Petri Net. Since the Occurrence Graph of a Petri Net model carries information on nodes as well as edges, the CTL extension in ASK-CTL allows specification of these properties with both state and transition information. For brevity, we refrain from presenting details on CTL. Interested readers may refer to [20] for more information on it.

An ASK-CTL expression, as defined in [21], is a state or a transition formula. The definitions, taken from [21], for both the categories are as follows:

Definition (ASK-CTL State Formula):- An ASK-CTL state formula A is defined by the following Backus Naur Form:

$$A ::= tt \mid \neg A \mid A \vee A \mid A \wedge A \mid \mathrm{EU}(A, A) \mid \mathrm{AU}(A, A) \mid \alpha \mid \langle B \rangle$$

where tt is interpreted as true, α is a function mapping from markings to Boolean values, B is a transition formula (defined below), U (until) is the standard temporal operator, A and E (for-all and exist respectively) are the path quantifiers. The temporal operator U is used in combination with one of the path quantifiers A and E, e.g. the formula $\mathrm{EU}(A_1, A_2)$ expresses the existence of a path from a given state (marking) where A_1 holds to a state where A_2 holds. Similarly, the formula $\mathrm{AU}(A_1, A_2)$ requires the property to hold along all paths from a given initial state [21].

Definition (ASK-CTL Transition Formula):- An ASK-CTL transition formula B is defined by the following Backus Naur Form:

$$B ::= tt \mid \neg B \mid B \vee B \mid B \wedge B \mid \mathrm{EU}(B, B) \mid \mathrm{AU}(B, B) \mid \beta \mid \langle A \rangle$$

where β is a function mapping from transitions to Boolean values and A is a state formula.

The functions α and β, mentioned in the formulae, are required to be implemented in Standard ML programming language [22]. They return a Boolean value after checking some property of the state and transition, respectively. In addition to the above mentioned operators, the ASK-CTL library offers a number of other derived operators to construct complex formulae for checking properties of a system. A list of some of these operators, along with their description, is provided in Table 4.

Table 4: Derived Operators

Operator	Description
Pos $(A) \equiv$ EU(tt, A)	It is *possible* to reach a state where A holds.
Inv $(A) \equiv \neg$Pos $(\neg A)$	A holds in every reachable state; A is *invariant*.
Ev $(A) \equiv$ AU(tt, A)	For all paths, A holds within a finite number of steps; A is *eventually* true.
Along $(A) \equiv \neg$Ev $(\neg A)$	There exist a path which is either infinite or ends in a dead state, along which A holds in every state.
$<B> A \equiv <B \wedge <A>>$	There exist an immediate successor state M satisfying A and B holds on the transition between the current state and M.
EX$(A) \equiv <tt> A$	There exists an immediate successor state in which A holds. Read 'Exist Next'.
AX$(A) \equiv \neg$EX$(\neg A)$	A holds in all immediate successor states, if any. Read 'For All Next'.

With this brief introduction to ASK-CTL, the model checking problem using ASK-CTL can be described as:

> Given an Occurrence Graph of a Petri Net and an ASK-CTL property (formula) p, determine if p holds in the initial state of OG, i.e. if OG $\models p$.

As per this definition, a Petri Net model is said to satisfy a property if the ASK-CTL formula that describes the property can be shown to hold in the initial state of the OG corresponding to the Petri Net being examined. The complexity of the ASK-CTL model checking algorithm, as implemented in CPN Tools (a tool for creating, simulating and analyzing Petri Nets) [23], is linear in the product of the size of formula and the size of the state space – $O(N(|V| + |E|))$, where N is the length of the formula, $|V|$ is the number of nodes (or states) and $|E|$ is the number of edges in the Occurrence Graph.

Using ASK-CTL for Verification of Inconsistency and Redundancy

In this section, we present the use of ASK-CTL for verification of inconsistent and redundant rules. The technique uses ASK-CTL formulae, representing certain behavioral properties, to look for patterns in an Occurrence Graph that correspond to redundancy and inconsistency. Our approach requires that circular instances of rules should be identified and isolated prior to applying model checking algorithm on an Occurrence Graph. This can be achieved by the application of structural analysis technique proposed in [2]. This is due to the fact that the presence of circular instances of rules in a Petri Net will result in cycles within the corresponding

Occurrence Graph, which may lead to an infinite execution of the model checking algorithm, when applied to the Occurrence Graph of such a Petri Net.

The construction of an Occurrence Graph, as mentioned in its definition, depends on the initial marking M_0 of a Petri Net. The working memory of the Production System representing a set of rules can be used to mark the conditions (i.e., places) that correspond to an initial marking of the Petri Net. An Occurrence Graph can then be constructed with the marked Petri Net representation of the rule set. It is assumed for this approach that the sets of mutually exclusive concepts μ are also provided along with the set of rules. Since the presented approach does not have the provision of identifying those sets by itself (e.g., a proposition and its negation), the user has to explicitly provide that information in addition to the rule set. The provided sets of mutually exclusive concepts will be used in identification of inconsistent cases. Examples of mutually exclusive sets from the sample rules are given below.

$$\mu_1 = \{P6, P7\}$$
$$\mu_2 = \{P13, P15\}$$

The rest of the section presents the ASK-CTL formulae for verification of redundancy and inconsistency and provides an explanation as to what they mean. It also shows the soundness and completeness of both the formulae, i.e., they can be used to capture only and every instance of the two cases.

(a) Verification of Inconsistency

Given an Occurrence Graph of a Petri Net with an initial marking M_0, the following ASK-CTL formula can be used to verify if the rule set, represented by the Petri Net, has inconsistent cases in it.

$$\text{Pos}\ (\alpha_1) \wedge \text{Pos}\ (\alpha_2) \qquad \text{(ex 1)}$$

where

α_1 is a state function that checks if u is true,
α_2 is a state function that checks if v is true,
such that $\{u, v\} \in \mu$

An implementation of the above ASK-CTL formula is shown in Table 5. It should be noted that the Pos function (in Table 5) is a pre-defined function in ASK-CTL library. A programmer, therefore, does not need to explicitly define the said function in the program that implements the above ASK-CTL formula. The Pos function takes a function name as an input parameter and calls the given function for all the states in the state space in an iterative fashion. A true value is returned from it on a true return value from the called function (during any iteration); otherwise a false value is returned from it at the end.

Description: It is possible that in the entire state space, two mutually exclusive concepts belonging to a set μ_i are true.

Soundness: It has already been mentioned that a rule set free of inconsistent rules will never result in conflicting knowledge. In other words, it can be said that the Occurrence Graph generated for such a conflict-free rule set will never have two mutually exclusive places from a set μ_i as marked. This implies that the ASK-CTL formula in the example (ex 1) does not capture

any case other than inconsistency, since it checks for the truth values of the mutually exclusive concepts (propositions) in the state space.

Completeness: The marked places in the states of an Occurrence Graph correspond to the propositions that are true in a certain given situation. Since the list of mutually exclusive concepts is maintained in this approach, the ASK-CTL formula in the example is checked for every mutually exclusive set in an Occurrence Graph. This guarantees that every possible conflicting case that can occur in a certain given situation will be identified. It is imperative that the list of mutually exclusive concepts be exhaustive and complete.

Table 5: An Implementation of ASK-CTL Formula for Inconsistent Case

```
{stateOne and stateTwo shown below are
global variables, which are used to store the
two state numbers that satisfy the
inconsistency property; whereas u and v are
variables that are set to two mutually
exclusive concepts in a set.}

var stateOne, stateTwo : integer;
var u, v : string;

function main(var mutuallyExclusiveSet :
                        array [1..2] of
string)
    var asketlResult : boolean;

    u := mutuallyExclusiveSet[0];
    v := mutuallyExclusiveSet[1];
    asketlResult := Pos("α1") & Pos("α2");

    if asketlResult = true
       print IncomingArcs of
states[stateOne];
       print IncomingArcs of
states[stateTwo];
       {Displaying Rules that cause
inconsistency.}
    else
       print "No inconsistency found";
    endif

end main
```

```
function Pos(var functionName : string)

    for i := 1 to length of states

       if functionName(i) = true

          return true;

       else

          continue;

       endif

    endfor


    return false;

end Pos

function α1 (var stateNum : integer)
    if marking of u in states[stateNum] > 0
       stateOne := stateNum;
       return true;
    else
       return false;
    endif
end α1

function α2 (var stateNum : integer)
    if marking of v in states[stateNum] > 0
       stateTwo := stateNum;
       return true;
    else
       return false;
    endif
end α2
```

(b) Verification of Redundancy

Given an Occurrence Graph of a Petri Net with an initial marking M_0, the following ASK-CTL formula can be used to identify if the rule set, represented by the Petri Net system, has redundancy.

$$\text{Pos}(\alpha_1 \wedge \text{Pos}(\alpha_2) \wedge (\alpha_3)) \qquad \text{(ex 2)}$$

where

α_1 is a state function that checks if state i has multiple output paths,
α_2 is a state function that checks if state j has multiple reachable paths from state i, and
α_3 is a function that checks if there are at least two disjoint sets of execution sequences between i and j.

An implementation of the ASK-CTL formula presented above (ex 2) is provided in Table 6.

Table 6: An Implementation of ASK-CTL Formula for Redundant Case

```
{Variable setofPaths (shown below) is a
structure that stores all paths from
stateOne to stateTwo, whereas
distinctPathsSet is a subset of setofPaths
which stores distinct paths from
stateOne to stateTwo. stateOne and
stateTwo are variables that are set to the
two state numbers that satisfy the
redundancy property.}

var stateOne, stateTwo : integer;
var setofPaths, distinctPathsSet : list;

function main()
    var asketlResult : boolean;

    asketlResult := Pos("α1" &
Pos("α2") & α3());
    if asketlResult = true
        print distinctPathsSet;
    else
        print "Redundancy not found";
    endif
end main

function Pos(var functionName : string)

    for i := 1 to length of states

        if functionName(i) = true
```

```
function α2 (var stateNum : integer)
    var parentNodes := getParentNodes of
states[stateNum];
    var numReachablePaths := 0;
    var path : list;
    for i := 1 to length of parentNodes
            if parentNodes[i] is reachable
from stateOne
                numReachablePaths++;
                path := getPath from stateOne
through
                        parentNodes[i] to
stateNum;
                add path to setofPaths;
            endif
    endfor


    if numReachablePaths > 1
            stateTwo := stateNum;
            return true;
    else
            return false;
    endif
end α2


function α3 ()

    for i := 1 to length of setofPaths

            for j := i+1 to length of
```

```
            return true;
        else
            continue;
        endif
    endfor

    return false;
end Pos

function α1 (var stateNum : integer)
    var outgoingArcs :=
getOutgoingArcs of
states[stateNum];
    if length of outgoingArcs > 1
        stateOne := stateNum;
        return true;
    else
        stateOne := 0;
        return false;
    endif
end α1
```

```
setofPaths
                if setofPaths[i] =
setofPaths[j]
                    break;
                endif
            endfor
            if j > length of setofPaths
                add setofPaths[i] to
distinctPathsSet;
            endif
        endfor
        if length of distinctPathsSet > 1
            return true;
        else
            return false;
        endif
end α3
```

Description: It is possible to reach a state that has multiple output paths from which there exists another reachable state having multiple input paths such that at least two of these paths between them are distinct (i.e. the execution sequences have no transition in common).

Soundness: The pattern for redundancy has already been presented in an earlier Section, which describes it as a set of multiple distinct paths from one state to another state within an OG. In fact, such a composition within an OG only corresponds to a redundant case. This implies that the ASK-CTL formula (given by ex 2) is guaranteed to capture only redundant cases, since it looks for such patterns of multiple distinct paths from one state to another state.

Completeness:- The ASK-CTL formula (given by ex 2) only identifies a single redundant case when applied to an Occurrence Graph. However, it is still possible to capture the entire set of redundant cases. Since an ASK-CTL formula is required to be implemented in SML programming language [22], the programming constructs of looping structures can be used to capture the entire set of redundant cases. The idea is to place the formula within a looping structure (e.g. while, do while) that captures a new redundant case, not captured before, in an iterative fashion. Such a modification will, however, increase the time complexity of the process by a factor of I, where I is the total number of redundant cases in the rule set.

Application of ASK-CTL Formulae on Sample Petri Net

This section presents an example to illustrate the new proposed technique. It uses the rules from Maritime Laws. The input situation for the example is described by the following query, which has been taken from sample scenario.

> "What are the permissible actions that can be taken against a hostile act of pirate vessel located in territorial waters?"
>
> The given query can be formally represented as,
>
> $$P5 \wedge P9 \wedge P10 \wedge P14$$

The Petri Net representation for the sample Maritime Laws has already been presented (see Fig. 28). There was a circular instance of a rule present in the given Petri Net representation, which was identified by the static analysis approach. The given Petri Net representation was modified by removing the identified circular instance comprising of second rule from Rule 5(a). Tokens were added to the places of modified Petri Net that correspond to the given situation (i.e. P5, P9, P10 and P14). An Occurrence Graph of the Petri Net corresponding to the given marking was generated. The resulting Occurrence Graph is shown in Fig. 32.

The application of two formulae on the generated OG revealed the presence of inconsistency and redundancy as well as their cause in the set of rules for the given input. The identification of their cause, i.e., rules leading to such cases, was made possible programmatically by maintaining a data structure (e.g. list), within the user-defined state functions, which stores nodes and arcs information whenever a certain property is found to be satisfied in a state. It was required since the ASK-CTL formula, as it is implemented in CPN Tools [23], only returns a Boolean value indicating the presence or absence of a property in a system.

(a) Inconsistent Rules

The ASK-CTL formula to detect an inconsistent case was applied for every set of mutually exclusive concepts. The identified sets of mutually exclusive concepts have already been presented. The application of the formula for the two sets uncovered that Rule 5(b) and Rule 6(b) were applicable in the given situation, both leading to two conflicting conclusions (P13 and P15). The time complexity of the ASK-CTL formula for inconsistent case is $O(M.N(|S| + |R|)$ where M is the number of mutually exclusive sets provided, N is the size of the formula ($N=2$ in our case), S is the number of states, and R is the number of rules applicable in the given situation.

(b) Redundant Rules

Similarly, the ASK-CTL formula for redundant case was applied to identify the presence of redundant rules. There were two sets of redundant rules identified by it. The first set included Rule 2 and Rule 7, whereas the second set consisted of Rule 4(b) and Rule 6(b). The Occurrence Graph shown in Fig. 32 highlights the pattern corresponding to the first set of redundant rules identified by the new proposed technique.

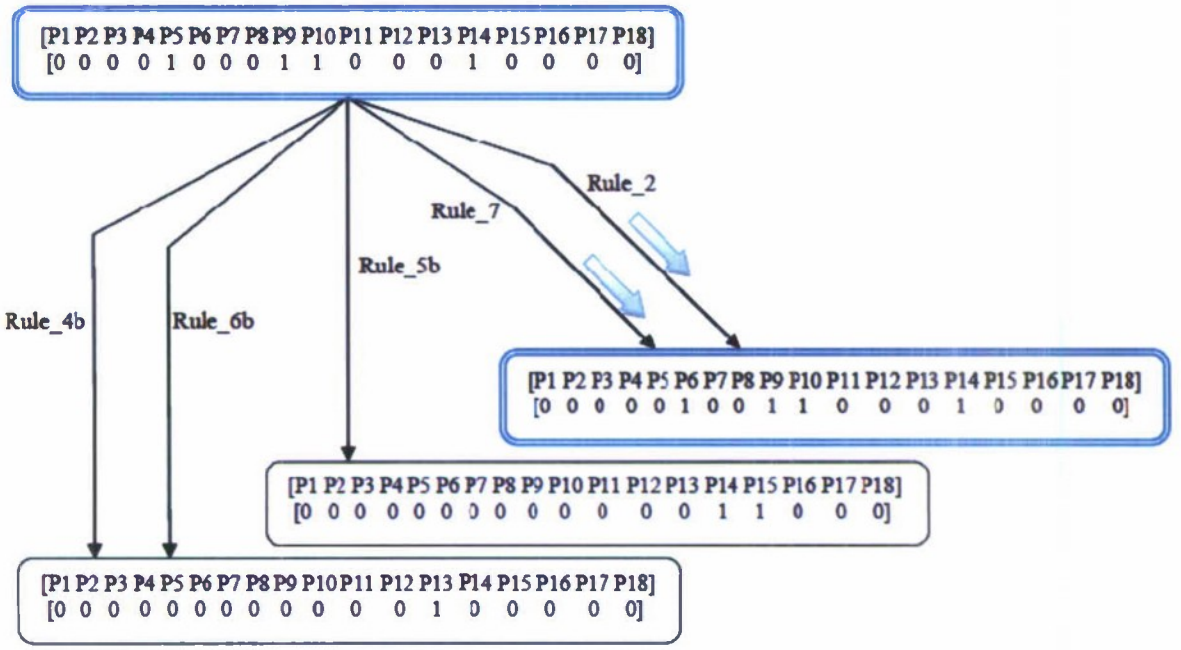


Fig. 32. Occurrence Graph corresponding to Modified Petri Net of Fig. 28

The complexity of the ASK-CTL formula for redundant case is $O(I(N(|S| + |R|) + L^2))$ where I is the number of redundant cases in the rule set, N is the size of the formula ($N=2$ in our case), S is the number of states, R is the number of rules applicable in the given situation, and L is the number of paths from the state (identified by α_1) to the state (identified by α_2).

4.8 Software Implementation

The approach presented in this section can be applied from within the CPN Tools [23]. It can also be implemented as a separate application by making use of an open source library, named Britney Suite [24]. The referred library is a set of functions written in Java that uses the CPN Tools simulator, making it possible to perform state space analysis in Java applications. Using the said library, we developed a utility named Rule Evaluation Routine or "RULER" to implement our proposed verification technique. It takes the file of formally represented rules as input and the set of facts, and mutually exclusive concepts from the user and generates a report on the identified set of cases. In addition to RULER, a suite of tools has also been developed to provide a computer-aided approach for the task of a Maritime Lawyer. The suite consists of three separate applications which provide the functionalities of management and verification of Maritime Laws as well as support decision-making.

Figure 33 shows the user interface of the Maritime Law Management System which has four basic sections (or panels). Section 1 displays the list (or collection) of Maritime Laws stored in the repository. The list of rules within the selected Maritime Law source is displayed in the area of Section 2. Selection of a rule shows its details in the area of Section 3, which include information about the title of the rule, its textual as well as logical representation, and the set of attachments and tags attached to it. Tags are keywords or phrases, created by the user, for cataloging (or indexing) the rules. A rule can have multiple tags attached to it. The set of tags,

attachments, and the two representations are assigned to the rules at the time they are entered in the system. Finally, Section 4 is the area that displays the entire list of tags assigned to the stored rules. The list of tags assists in searching the rules from the repository by enabling a user to select a tag or a combination of them, which extracts all the rules having those tags assigned to them. In other words, it can be said that they provide a more efficient search mechanism compared to other search procedures. For example, the tag of "piracy" on selection will pull all the rules having that tag assigned to them. The presented example is based on the assumption that the tag was created and assigned to rules from various Maritime Law sources that address piracy. Figure 34 presents another view of the first three sections, showing the interconnection between various data entities.

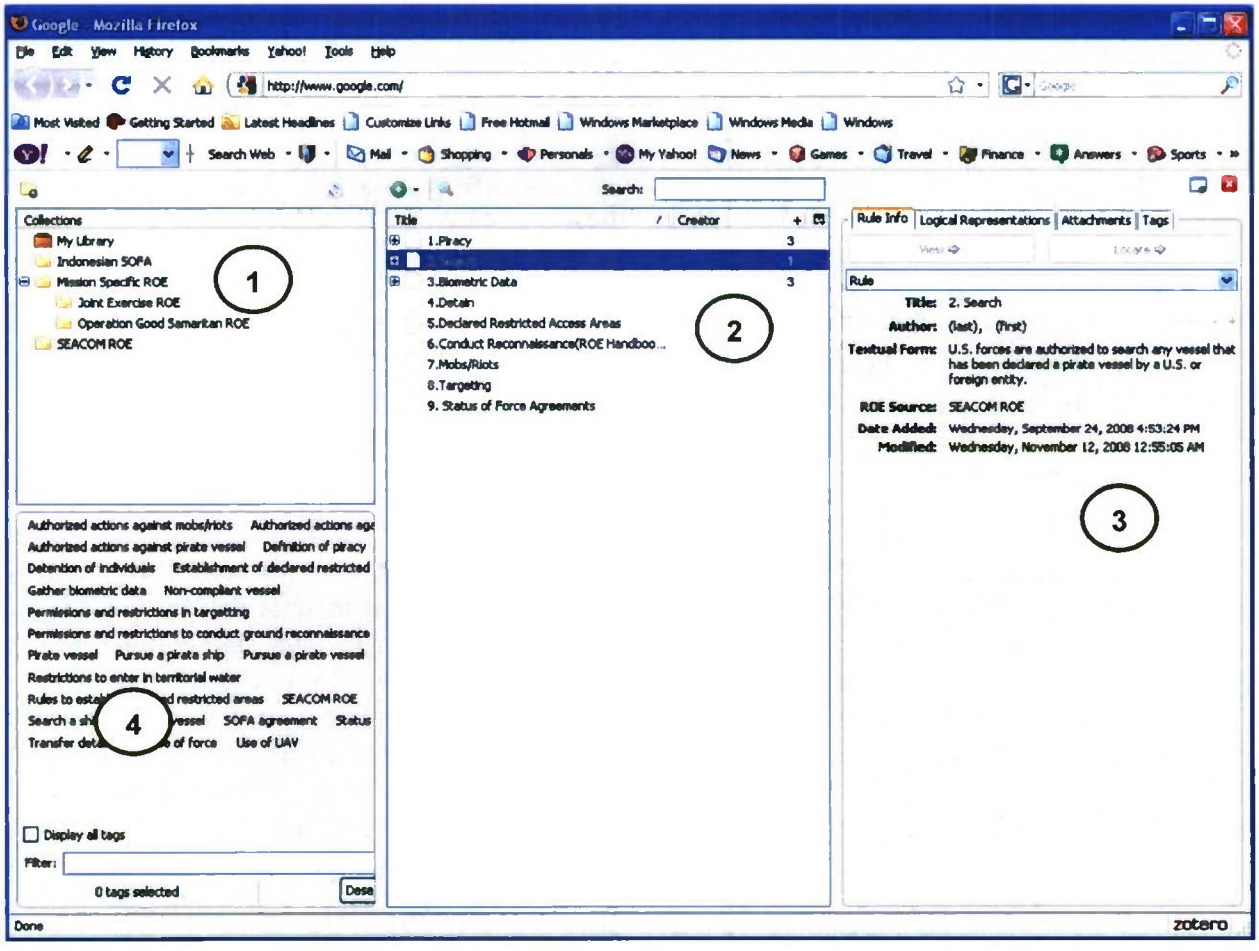


Fig. 33. User Interface of Maritime Law Management System

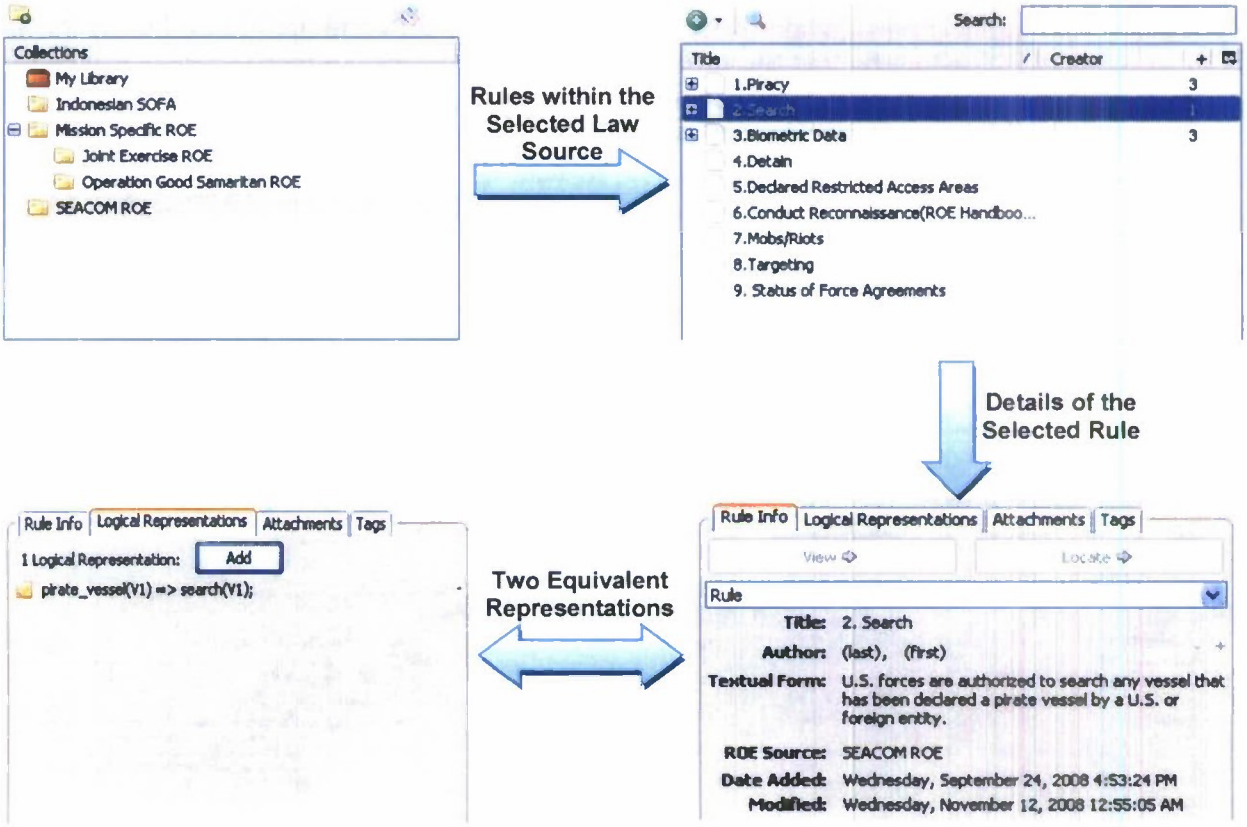


Fig. 34. Various Components within Maritime Law Management System

Users can enter information from various Maritime Law sources, which may include their names, rules within them, and their formal representation using additional features provided by the system, using the interface shown in Fig. 33. The screenshots in Fig. 34 show menus on the top side, represented by '+', and buttons that can be used to perform such operations. In addition to these features, the system offers other important functionalities for a Maritime Lawyer, which are stated below:

- It facilitates text-based searching, which can be used to search the rule-base by entering keywords or phrases, in addition to tag-based search mechanism.
- The selected set of rules, along with their textual as well as formal representations, can be exported as a file to other applications.
- A detailed report about the selected set of rules can be generated, which includes their source information and the two representations.

An example of a test case is presented below that uses the following query.

> "What are the permissible actions that can be taken against a hostile act of pirate vessel located in territorial waters?" (Rephrased Query 5, Philippines Tsunami Humanitarian Assistance Disaster Relief Mission: OPLAW/ROE Scenario v1.0)

Since the query refers to a situation in which a pirate vessel is encountered, a search was made in the rule-base by selecting the tag "Pirate Vessel". The selected tag extracted all the set of rules that talk about various actions, whether permitted or prohibited, against the pirate vessel. A

screenshot of the application showing the resulting rules, from different applicable law sources, is shown in Fig. 35. The highlighted Maritime Law, shown in the figure, represents the source of the selected rule. A detailed report of the resulting set of rules was also generated, a portion of which is shown in Fig. 36. It should be noted that the current implementation of the suite of tools uses "&" symbol for the conjunctive connective instead of "∧", which is used in the formal representation.

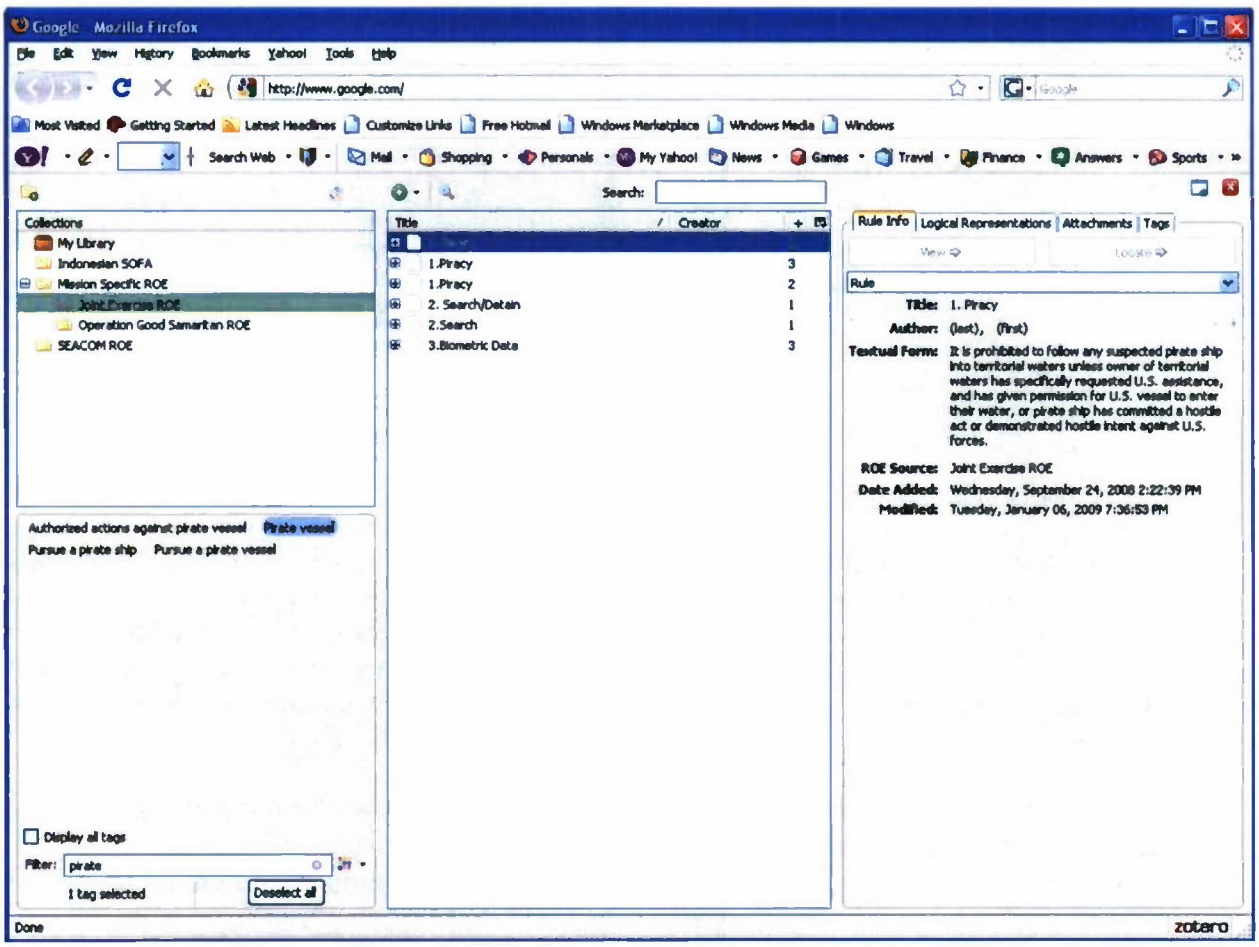


Fig. 35. GUI of Maritime Law Management System Showing Search Results

1. Piracy

Type Rule
Textual Form It is prohibited to follow any suspected pirate ship into territorial waters unless owner of territorial waters has specifically requested U.S. assistance, and has given permission for U.S. vessel to enter their water, or pirate ship has committed a hostile act or demonstrated hostile intent against U.S. forces.
ROE Source Joint Exercise ROE
Date Added Wednesday, September 24, 2008 2:22:39 PM
Modified Tuesday, January 06, 2009 7:36:53 PM

Logical Representation:

pirate_vessel(V1) & territorial_water(C1) & position(V1,C1) & entry_request(C1) & permitted(C1) => pursue(V1,C1);

pirate_vessel(V1) & territorial_water(C1) & position(V1,C1) & hostile_act(V1) => pursue(V1,C1);

2. Search/Detain

Type Rule
Textual Form Only ships that have been declared pirate ships by the policing entity may be stopped/boarded.
ROE Source Indonesian SOFA
Date Added Wednesday, September 24, 2008 1:47:31 PM
Modified Tuesday, January 06, 2009 7:43:11 PM

Logical Representation:

pirate_vessel(V1) => search(V1);

Fig. 36. Generated Report for the Selected Rules

Ruler (Rule Evaluation Routine)

The technique for formal verification of a set of rules, selected from various Maritime Laws, has already been presented in this report. The behavioral analysis in it, which addresses portion of the verification problem, is performed by CPN Tools. A utility named RULER for Rule Evaluation Routine was developed in Java to implement the proposed verification technique. It uses an open source library, named Britney Suite [35], to interact with CPN Tools. The referred library is a set of functions written in Java that uses CPN Tools simulator, making it possible to

integrate Petri Net simulation (i.e. Petri Net execution mechanism) and its state space analysis in Java applications. The implemented utility works by taking the file of the exported rule set (generated by Maritime Law Management System), the set of facts, and mutually exclusive concepts from the user and performs verification analyses using the given information, consequently, providing a report on identified inconsistent, redundant, incomplete, and useless rules. A Petri Net model is also generated by the application corresponding to the set of rules that can be used by the Rule Execution Engine, presented in the next section, to assist in decision-making process. The generation of Petri Net by RULER and its exportation to Rule Execution Engine is, however, transparent to the user. Figs. 37, 38 and 39 show some of the screenshots of RULER, displaying some results from the implemented verification technique.

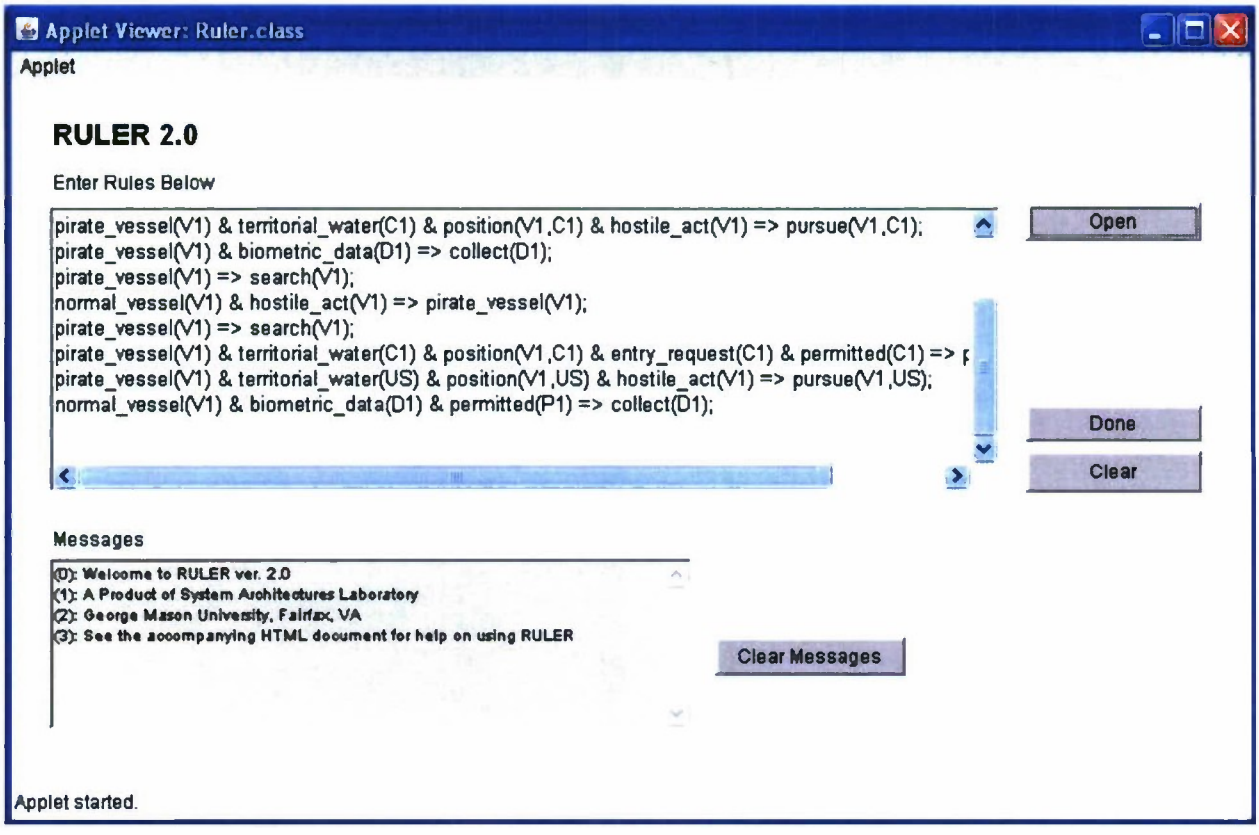


Fig. 37. Input Rules from Maritime Law Management System

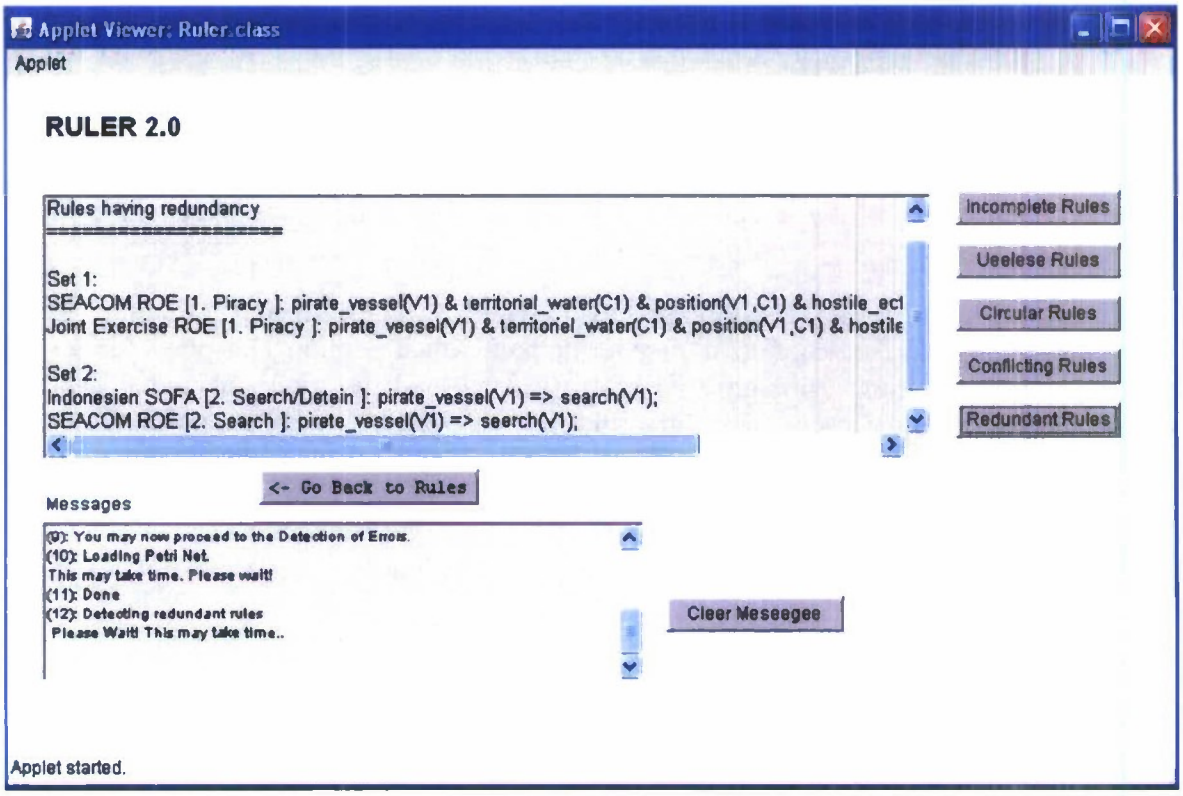


Fig. 38. Results showing Identified Redundant Cases

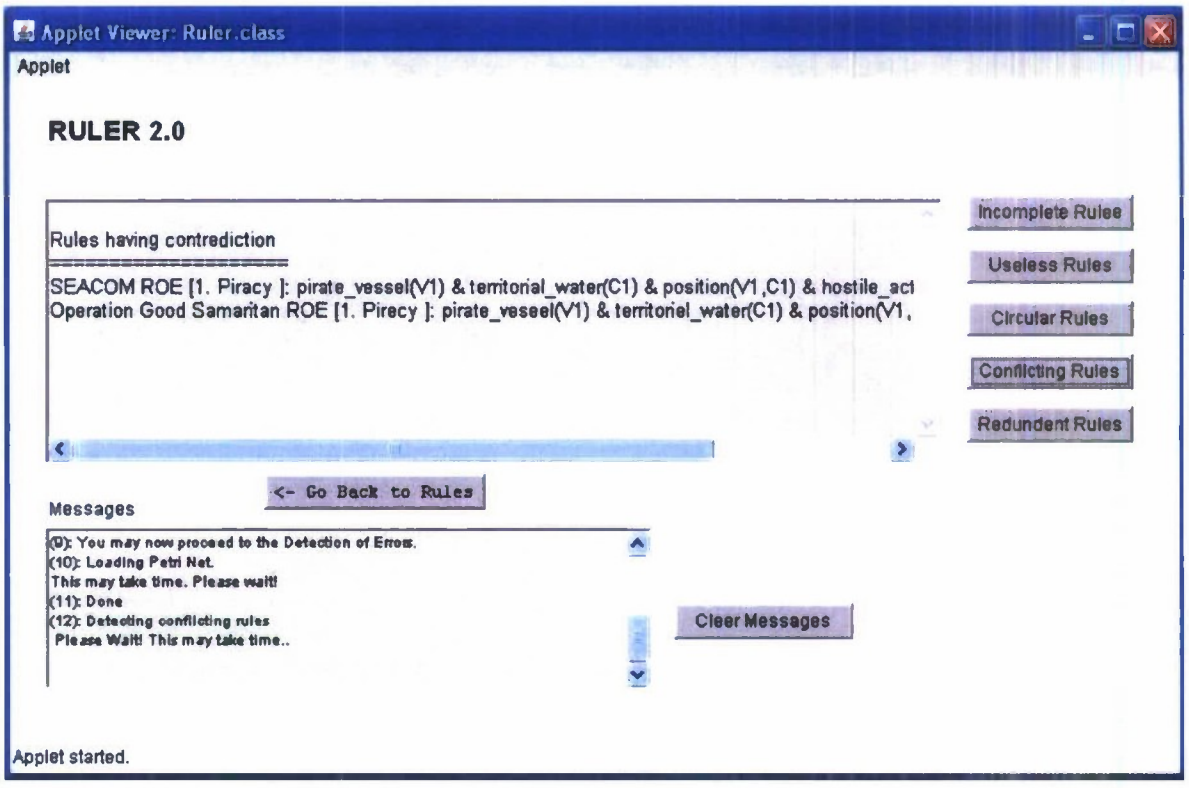


Fig. 39. Results showing Identified Inconsistent Cases

RulEx (Rule Execution Engine)

The reasoning mechanism for the OPLAW/ROE rule repository was incorporated into the suite by developing a reasoning engine known as "Rule Execution Engine (RulEx)". This application makes use of the Britney Suite library [35] to perform simulation on the Petri Net model that was generated by RULER. Some of the screenshots of Rule Execution Engine are presented in Figs. 38 and 39.

Figure 40 shows the results of rule execution process that derives a new set of information based on the given facts. The interface consists of five sections in which Section 1 displays the set of rules triggered, Section 2 shows the entire set of facts (input and resulting), Sections 3 and 4 correspond to the graphical representation of the rule set, and Section 5 presents the summary of the entire execution process.

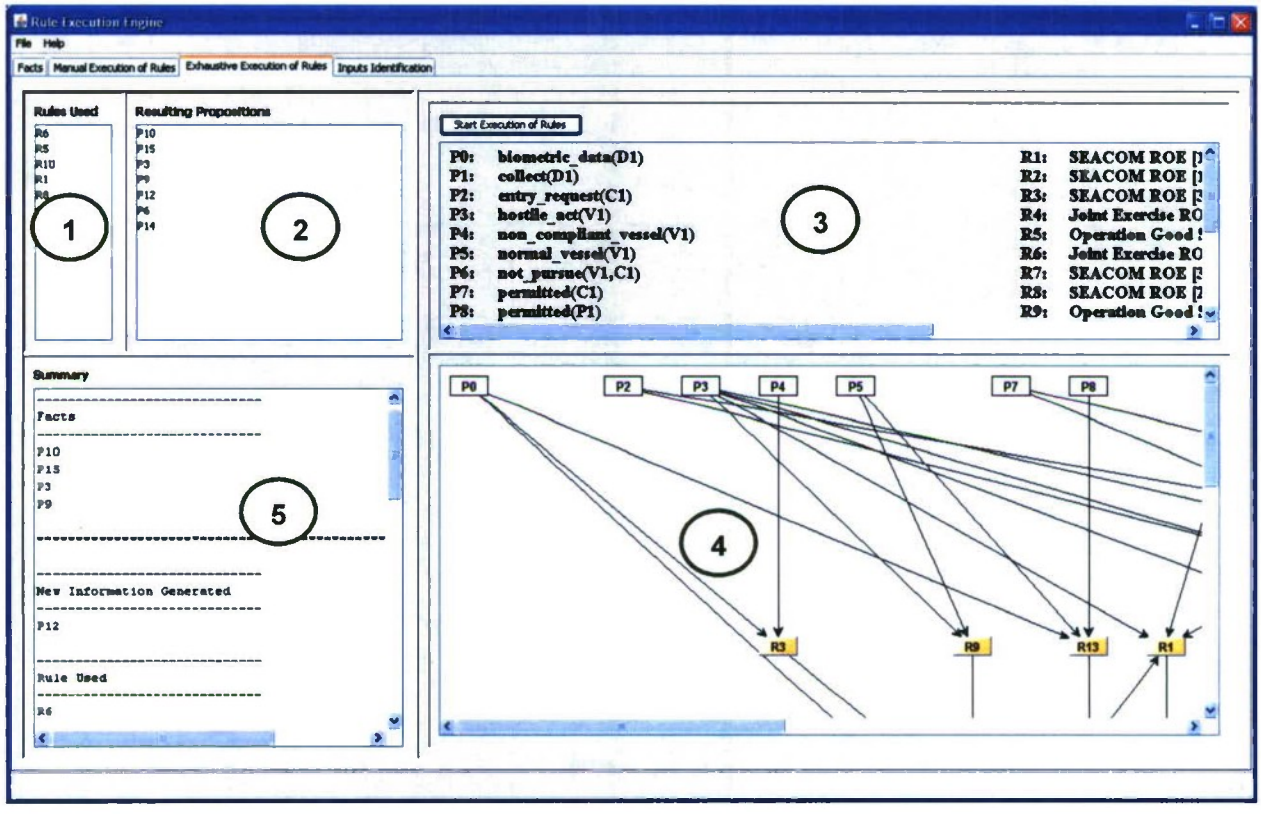


Fig. 40. Example of Rule Execution Process

Similarly, the screenshot in Fig. 41 presents the process of identifying input conditions that support a certain action. The selected concept (i.e., P12) in the top-left panel of the interface corresponds to the action proposition, whereas the bottom-left panel displays all the identified set of input conditions that support it.

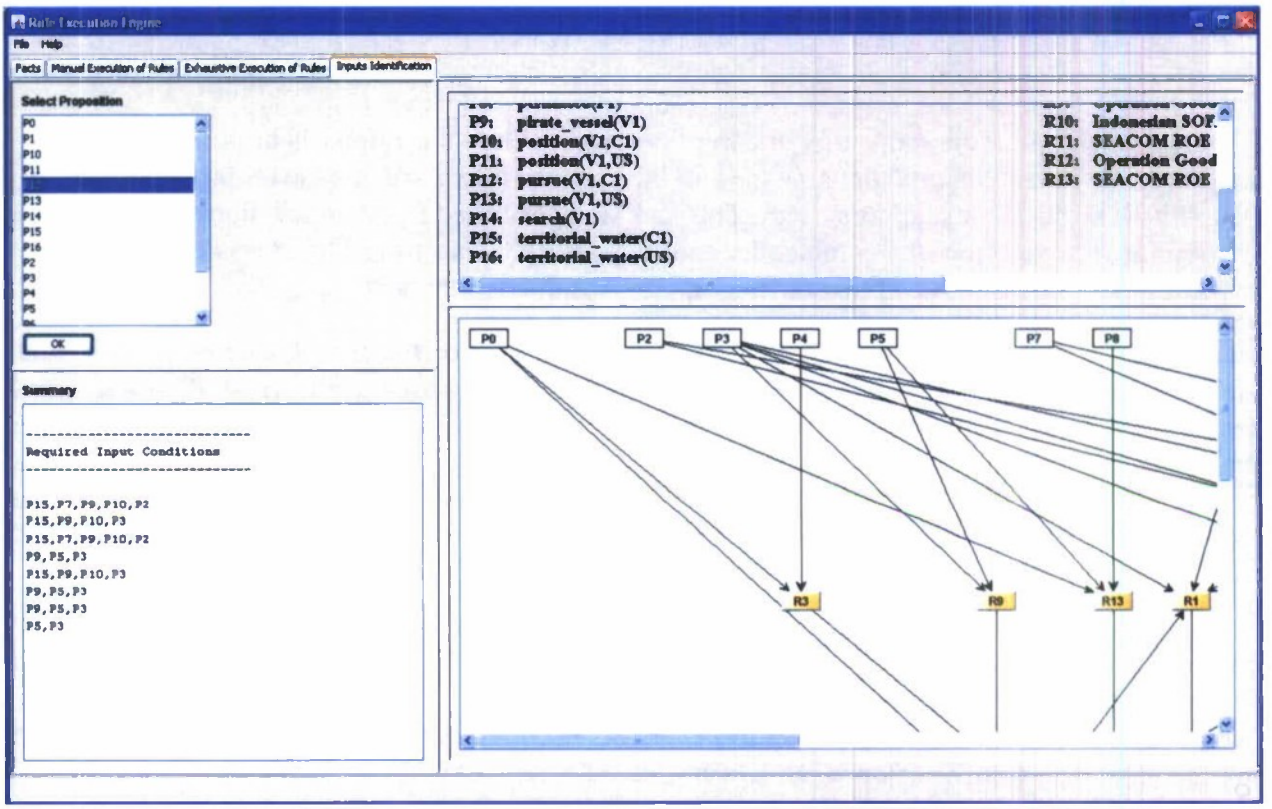


Fig. 41. Example of Input Conditions Identification Process

4.9 Summary

In this research task, GMU developed an approach that uses a formal model checking methodology to detect redundant and inconsistent cases in a set of rules selected from across different Maritime Laws. The approach can provide considerable help to a Maritime Lawyer, i.e., a Staff Judge Advocate (SJA) in deciding the set of legal actions from the applicable rules after careful consideration of the cases identified by it. The approach, implemented as a software suite of tools, provides a repository of OPLAW/ROE sets of rules from different law sources. The repository allows for an information foraging and compiling capability. The selected sets of rules are then analyzed by the RULER tool for consistency and redundancy. Finally, a reasoning engine provides an "If-then' analysis capability to a user for the final selection of applicable rules in given mission situation.

5. CONCLUSION

All tasks described in the scope of work were executed. While the emphasis in the first years of the project was on augmentation strategies, in the last years it was on the development of the OPLAW/ROE decision support system. This task was generated from observations made and debriefings received from the commander and staff of ESG ONE (the July 2005 deployment of Expeditionary Strike Group One) at the completion of the deployment.

The primary objective of the effort was to study and analyze the new concepts of the Navy Maritime Operations Center (MOC) focusing on the scalability aspects of the MOC concept and develop tools and techniques that could support the design and evaluation of augmentation strategies. Two basic forms of augmentation were developed: Augmentation by Staff and Augmentation by Function. In any particular case, either type or a combination of both would be appropriate. That will allow MOCs to change or adapt to new missions and new levels of command and control. .

A suite of tools for command and control organization design was upgraded and modified so that it can support the modeling and analysis of MOC scalability. Several workflows were developed that support the process of using the tools to enable the design of MOC augmentation strategies. Techniques for considering the impact of cyber contested environments and space limitations on command vessels were developed. However, this only began to address the issue of resilience of the architecture to cyber attacks. It set, however, the foundation for further work in this area. Scenarios were created and evaluated using the tools to demonstrate the tools and workflows.

In the OPLAW/ROE task earlier work on the validation and verification of rule bases was used in addressing the dynamically changing rules of engagement of a maritime force as it crosses different geographical areas. GMU developed an approach that uses a formal model checking methodology to detect redundant and inconsistent cases in a set of rules selected from across different Maritime Laws. The approach can provide considerable help to a Maritime Lawyer, i.e., a Staff Judge Advocate (SJA) in deciding the set of legal actions from the applicable rules after careful consideration of the cases identified by it. The approach, implemented as a software suite of tools, provides a repository of OPLAW/ROE sets of rules from different law sources. The repository allows for an information foraging and compiling capability. The selected sets of rules are then analyzed by the RULER tool for consistency and redundancy. Finally, a reasoning engine provides an "If-then' analysis capability to a user for the final selection of applicable rules in given mission situation. While the theoretical foundation was established and a prototype was developed, further work is needed to complete the design and test it with operators.

6. REFERENCES

[1] http://www.zotero.org/

[2] A. Zaidi and A. Levis, "Validation and verification of decision making rules," *Automatica*, vol. 33, 1997, pp. 155-169.

[3] M. D. Zisman, "Use of production systems for modeling asynchronous, concurrent processes," *ACM SIGART Bulletin*, 1977, p. 23.

[4] A. Giordana and L. Saitta, "Modeling production rules by Means of Predicate Transition networks," *Information Sciences*, vol. 35, 1985, pp. 1-41.

[5] D. Zhang and D. Nguyen, "A technique for knowledge base verification," *[Proceedings 1989] IEEE International Workshop on Tools for Artificial Intelligence*, Fairfax, VA, USA: , pp. 399-406.

[6] Du Zhang and Doan Nguyen, "A tool for knowledge base verification," in *Development of Knowledge-Based Shells*, World Scientific Publishers, 1992.

[7] N. K. Liu and T. Dillon, "An approach towards the verification of expert systems using numerical petri nets," *International Journal of Intelligent Systems*, vol. 6, 1991, pp. 255-276.

[8] R. Agarwal and M. Tanniru, "A Petri-Net based approach for verifying the integrity of production systems," *International Journal of Man-Machine Studies*, vol. 36, 1992, pp. 447-468.

[9] S. Yang, A. Lee, W. Chu, and Hongji Yang, "Rule base verification using Petri nets," *Proceedings. The Twenty-Second Annual International Computer Software and Applications Conference (Compsac '98) (Cat. No.98CB 36241)*, Vienna, Austria: , pp. 476-481.

[10] Xudong He, W. Chu, H. Yang, and S. Yang, "A new approach to verify rule-based systems using Petri nets," *Proceedings. Twenty-Third Annual International Computer Software and Applications Conference (Cat. No.99CB37032)*, Phoenix, AZ, USA: , pp. 462-467.

[11] S. Yang, J. Tsai, and Chyun-Chyi Chen, "Fuzzy rule base systems verification using high-level petri nets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, 2003, pp. 457-473.

[12] Z. Ding, M. Pan, C. Jiang, and Y. Han, "Using Petri nets to verify acyclic rule-based system," *Frontiers of Electrical and Electronic Engineering in China*, vol. 3, 2008, pp. 155-161.

[13] Qingfeng Wu, Changle Zhou, Jinlin Wu, and Chaonan Wang, "Study on Knowledge Base Verification Based on Petri Nets," *2005 International Conference on Control and Automation*, Budapest, Hungary:, pp. 997-1001.

[14] H. J. Levesque, "The logic of incomplete knowledge bases," in *On Conceptual Modeling: Perspective from Artificial Intelligence, Databases and Programming Languages*, M. L. Brodie, J. Mylopoulos and J. W. Schmidt (eds.), Springer-Verlag, New York, 1984, pp. 165-189.

[15] A. Ligeza, *Logical Foundations for Rule-Based Systems*, Berlin: Springer, 2006.

[16] S. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach*, Upper Saddle River N.J.: Prentice Hall/Pearson Education, 2003.

[17] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*, Cambridge Mass.: MIT Press, 1999.

[18] A. K. Zaidi and A. H. Levis, "Verification of System Architectures Using Modal Logics and Formal Model Checking Techniques," Conference on Systems Engineering Research (CSER), 2006, Los Angeles, CA.

[19] W. Reisig and G. Rozenberg, *Lectures on Petri nets: advances in Petri nets*, Berlin; New York: Springer, 1998.

[20] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," ACM Transactions on Programming Languages and Systems, vol. 8, 1986, pp. 244-263.

[21] A. Cheng, S. Christensen, and K. H. Mortensen, "Model checking coloured petri nets exploiting strongly connected components," Proceedings of the International Workshop on Discrete Event Systems, WODES96. Institution of Electrical Engineers, Computing and Control Division, Edinburgh, UK, 1996.

[22] R. Milner, M. Tofte, R. Harper, and D. MacQueen, *The Definition of Standard ML: Revised*, Cambridge Mass.: MIT Press, 1997.

[23] "CPN Tools: Computer Tool for Coloured Petri Nets", University of Aarhus, http://wiki.daimi.au.dk/cpntools/cpntools.wiki

[24] "Britney Suite: Experimental Test-bed for New Features for CPN Tools", University of Aarhus, http://wiki.daimi.au.dk/britney/britney.wiki

Pergamon

PII: S0005-1098(96)00165-3

Automatica, Vol. 33, No. 2, pp. 155–169, 1997

0005-1098/97 $17.00 + 0.00

Validation and Verification of Decision Making Rules*

ABBAS K. ZAIDI† and ALEXANDER H. LEVIS†

A methodology based on Petri nets for addressing the general problem of detecting problematic cases in a set of rules expressed as statements in formal logic is presented and illustrated through an example.

Key Words—Decision making; rule-based systems; Petri nets.

Abstract—A methodology for the validation and verification of decision making rules is presented. The methodology addresses the general problem of detecting problematic cases in a set of rules expressed as statements in formal logic. This representation of the decision rules makes the problem general in terms of application domains, and also provides an analytical base for defining errors. The approach is based on viewing a rule base as an organization of information that flows from one process (rule) to another. The key step in the methodology is the transformation of the set of decision rules into an equivalent Petri net. The static and dynamic properties of the resulting Petri net are shown to reveal patterns of structures that correspond to the problematic cases. The techniques presented in this paper are based on theory and are supported by software tools.

1. INTRODUCTION

The decision making process that an organization carries out can often be represented by a set of rules. An example of such an organization could be one that obtains inputs from a set of sensors, identifies a task based on these inputs and, on the basis of attributes of the task, determines a response. The set of decision rules may have been derived from a theory (normative or prescriptive) or they may have been obtained from empirical studies (descriptive) including knowledge elicitation from domain experts. The problem of dynamic *task allocation* in team decision making requires the partitioning of these rules across the individual decision makers in the organization. The manner in which these rules are obtained and the process of partitioning the rule base across decision making entities (human and machine) can introduce inconsistencies, incompleteness and redundancies, as well as problems in coordination. Consequently, there is a clear need for the validation and verification (V&V) of rule bases. The following example illustrates these issues.

A generic naval Tactical Command Center (TCC) (Levis *et al.*, 1995) provides an example of an organization where several decision makers are required to make certain decisions before an overall response of the system can be executed. Suppose that five decision makers, with corresponding responsibilities, constitute a hypothetical TCC (Table 1). The organization receives sensory inputs from a radar and a sonar sensor, and non-sensory inputs from certain databases and intelligence reports/updates. The following alphabets represent these sources of inputs.

Radar: {r1, r2,r3};
Database: {db1, db2, . . .};

Sonar: {p1, p2, p3};
Intel: {intel1, intel2, . . .}.

The decision process of TCC can be modeled as a set of rules that maps combinations of these inputs to the output responses:

Output: {Fire-SSM, Fire-SAM, Fire-Torpedo, Fire-Depth-Charge, No-Action, . . .}.

The following is an example of a decision rule that could apply during a training exercise (much more complex rules apply in peacetime):

(*R*1)	IF	the input from radar is r1
	and	the input from sonar sensor is p2
	and	the intelligence update is intel4
	and	the database information is db2
	THEN	the incoming object is an enemy vessel
	and	Fire-Torpedo

The task allocation requires that rule R1 be assigned to a decision maker who not only has

* Received 5 August 1995; received in final form 17 July 1996. The original version of this paper was presented at the 6th IFAC Symposium on Analysis, Design and Evaluation of Man–Machine Systems, which was held in Cambridge, MA. U.S.A. during 27–29 June 1995. This paper was recommended for publication in revised form by Editor A. P. Sage. Corresponding author Professor A. H. Levis. Tel. +1 703 993 1619; Fax +1 703 993 1706; E-mail alevis@gmu.edu.

† School of Information Technology and Engineering, George Mason University, Fairfax, VA 22030, U.S.A.

Table 1. Responsibilities of decision makers

Decision maker	Responsibilities
Radar Operator, RO	Monitors air and surface threats
Sonar Operator, SO	Monitors surface and subsurface threats
Anti-Air Warfare Commander AAWC	Controls missiles against air and surface threats
Anti-Sub/Surface Warfare Commander ASWC	Controls depth-charges, and torpedoes against surface and subsurface threats
Commander CC	Receives data from RO and SO, and sends commands to AAWC and ASWC

access to all the inputs of R1 but should also have the capability to carry out the output response—'Fire-Torpedo'. The rule requires four inputs: r1 and p2 are sensory information from RO and SO respectively, while intel4 and db2 are inputs available only to the commander (CC). On the other hand, the output response 'Fire-Torpedo' can only be carried out by the ASWC. In the TCC organization, where different decision makers have access to different information sources and can execute different responses, a rule like R1 cannot be assigned to a single decision maker. Instead, R1 has to be decomposed into several smaller rule sets with the same cumulative effect as R1, and then each smaller rule set is assigned to a different decision. A possible decomposition and assignment of R1 is as follows:

($R1_1$) (*Rule assigned to RO*)
IF the input from radar is r1
THEN the assessed information R1 sent to the CC

($R1_2$) (*Rule assigned to SO*)
IF the input from sonar sensor is p2
THEN the assessed information P2 sent to the CC

($R1_3$) (*Rule assigned to CC*)
IF assessed information from RO is R1
and assessed information from SO is P2
and the intelligence update is intel4
and the database information is db2
THEN the incoming object is an enemy vessel
and ASWC is commanded to attack

($R1_4$) (*Rule assigned to* ASWC)
IF ASWC is commanded to attack
THEN Fire Torpedo

In this illustrative example, the decomposition of $R1$ adds several new rules to the organization's rule base. These added rules do not affect the functionality of the decision process; instead, they facilitate the team decision making by incorporating the coordination requirements among the team members. (One can also think of decomposition as the process where a complex and difficult problem is solved by solving several smaller and easier problems.) Although the decomposition presented seems feasible, it has introduced a potential source of error in the organization's decision process: consider the decomposition of another similar rule, and a consequent addition of the following two more rules to the rule base.

($R2_3$) (*Rule assigned to CC*)
IF assessed information from RO is R2
and assessed information from SO is P3
and the intelligence update is intel4
and the database information is db3
THEN the incoming object is an enemy submarine
and ASWC is commanded to attack

($R2_4$) (*Rule assigned to ASWC*)
IF ASWC is commanded to attack
THEN Fire Depth-Charge

The potential error introduced as a result now becomes apparent, since the two rules $R1_4$ and $R2_4$ pose an ambiguity to the ASWC as to what weapon system to fire once a command-to-attack is received from the CC. The source of error is the incomplete information conveyed to the ASWC by the CC. This example illustrates how a rule that is not erroneous may suddenly become incomplete, ambiguous and/or inconsistent in the presence of other rules. In a TCC, such erroneous cases may result in the execution of the wrong response with possibly catastrophic consequences, i.e. friendly fire, firing wrong weapon, no action against a threat, etc. More generally, even if the physical and coordination structures of an organization are feasible, the presence of such problems in the set of rules assigned to decision making entities can result in poor performance and unreliable system response.

Several techniques and methods for the validation and verification of rule bases have been reported in the literature, especially by the Expert Systems (ES) community. Lee and O'Keefe (1994) classified recently developed V&V tools and techniques into several categories depending upon the approach taken, i.e. analytical modeling, human support, running of test cases against set standards, etc. However, a majority of the V&V tools are based on running test cases through a system and comparing results against known results or expert opinion (O'Keefe *et al.*, 1987). Examples of this simple approach range from MYCIN's early validation

(Yu *et al.*, 1979) to the recently reported validation tool SAVES (Smith and Kandel, 1993). On the other hand, a very small number of reported validation techniques are available that use quantitiative validation (Lehner, 1989; Smith and Kandel, 1993). The quantitative validation approach uses statistical methods to compare expert system's performance against either test cases or human expert.

In addition to these approaches, a number of researchers have proposed several tools that verify a system's correctness (defined differently for different systems) by detecting and identifying logical and structural erors hidden in the expert system. A number of techniques are based on transforming the elements of an expert system into another analytically sound domain, i.e. predicate/transition nets (Zisman, 1978; Giordana and Saitta, 1985; Zhang and Nguyen, 1989, 1992; Liu and Dillon, 1991; Agarwal and Tanniru, 1992), graph models (Lim *et al.*, 1986; Wilkins and Buchanan 1986), decision tables and dependency charts (Cragun and Steudel, 1987; Nguyen *et al.*, 1987), KB-reduction (Ginsberg, 1988), Boolean techniques (Charles and Dubois, 1991), etc.

The mathematical tools available for the domain are then applied to this transformed expert system to uncover hidden erroneous and problematic cases. In addition, a number of other V&V tools and techniques are also reported (Davis, 1976; van Melle, 1981; Suwa *et al.*, 1985; Stachowitz and Comb, 1987; Stickel, 1988; Ayel and Laurent, 1991; De Raedt *et al.*, 1991; Smith and Kandel, 1993) in the literature. However, all the reported cases have at least one of the following weaknesses:

(a) based on running test cases;

(b) restricted in terms of implementation, representation of rules, types of problems handled, and constraining requirements for the applicabiilty of the approach;

(c) errors defined between pairs of rules or among rules in a subset of the entire rule base;

(d) combinatorial enumeration required to solve problems.

Boehm (1984), Lee and O'Keefe (1994) and Zaidi (1994) provide condensed bibliographies of the related work in verification and validation of expert systems. A collection of some of the recent approaches in V&V of expert systems is compiled in Gupta (1991) and Ayel and Laurent (1991).

A new approach to the general problem of detecting problematic and erroneous cases in a set of rules is proposed that overcomes the limitations of the existing methods. The methodology requires that the rules be expressed as statements in formal logic. An algorithm is then used to transform these conditional statements to an equivalent Petri net (PN) representation. The transformation is based on a mapping between the logical operations of conjunction ($\wedge$), disjunction ($\vee$), and implication and the notions of synchronization, concurrency, choice, etc. of Petri net theory. This visualization of a rule base in terms of an organization of interacting processes opens a wealth of analytical tools and techniques that have been developed by researchers and system analysts to perform structural and dynamic analyses on PNs. Once a rule base is transformed into a PN, the solution to the problem becomes a direct application of these analytical tools of Petri net theory. The V&V of the rule base is done first by exploiting the structural properties of the Petri net representation and then by constructing the occurrence graph directly from the Petri net representation.

In the next section, the problem of task allocation is presented by first considering a hierarchical organization of five decision makers (DM). An initially 'correct' rule base is then decomposed and partitioned so that the rules can be assigned to the individual DMs. The type of problems that can find their way into the decomposed rule base are introduced in the following section. The decomposed rule base is then checked for inconsistencies, incompleteness, and redundancies by the proposed methodology. The Petri net transformation of the rule base is presented in Section 4. The results of the static and dynamic analyses performed on the Petri net representation are presented in Sections 5 and 6 respectively.

2. PROBLEM DEFINITION

The example set of organizational decision rules illustrated in this section was motivated by the 'message puzzle task (MPT)' of Wesson and Hayes-Roth (1980). The MPT involved a game-like environment in which words and phrases move about in a two-dimensional grid that resembles a puzzle board. A group of players, each of whom can see a portion of the grid, must communicate among themselves to identify the moving items and eventually solve the puzzle. In the illustration presented in this section, a 4×3 grid (consisting of 12 cells) representing the puzzle board is considered (Fig. 1). In contrast to the MPT experiment, the messages on the grid do not move but appear on

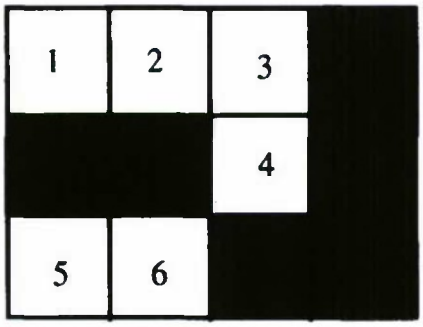


Fig. 1. An instance of the grid.

certain cells in the grid. The messages consist of letters from an input alphabet of integers, where each integer represents an event. Based on the appearance of these messages in certain cells, the set of rules infers a sequence of events out of a possible three sequences in which these events can occur.

The set of all events is given as E = {1, 2, 3, 4, 5, 6}, where each integer represents the occurrence of an event. The appearance of an integer (from set E) in one of the cells of the grid is considered as the basic input to the set of decision rules. The following 12 basic inputs are identified:

$$U = \{P1, P2, P3, \ldots, P12\}$$

where the proposition symbols P1–P12 represent the following information from the grid:

P1: integer 1 in sector 1
P2: integer 2 in sector 2
P3: integer 3 in sector 3
P4: integer 1 in sector 4
P5: integer 2 in sector 5
P6: integer 3 in sector 6
P7: integer 4 in sector 7
P8: integer 4 in sector 8
P9: integer 5 in sector 9
P10: integer 6 in sector 10
P11: integer 5 in sector 11
P12: integer 6 in sector 12

Based on these inputs, the rule base *RB*1 tries to interpret the inputs in terms of three possible outcomes (sequence of events), which are characterized as the main concepts of the rule base:

$$\Psi = \{A, B, C\} \qquad (2)$$

where

A: sequence of events is 4, 5, 6, 3, 2, 1
B: sequence of events is 1, 2, 3, 4, 5, 6
C: sequence of events is 6, 5, 4, 3, 2, 1

*Rule Base, RB*1.

Rule1: $P1 \wedge P2 \wedge P3 \rightarrow Q1$
Rule2: $P1 \wedge P2 \wedge P6 \wedge P7 \rightarrow R2$
Rule3: $P4 \wedge P5 \rightarrow Q3$
Rule4: $P3 \wedge Q3 \rightarrow R3$
Rule5: $P4 \wedge P5 \wedge P9 \wedge P12 \rightarrow R3$
Rule6: $R2 \wedge R4 \rightarrow C$
Rule7: $P7 \wedge P9 \wedge P10 \rightarrow Q2$
Rule8: $R3 \wedge R1 \wedge P8 \wedge P9 \rightarrow A$
Rule9: $P8 \wedge P11 \wedge P12 \rightarrow R4$
Rule10: $Q1 \wedge Q2 \rightarrow B$
Rule11: $P6 \wedge P7 \rightarrow R1$

where

Q1: partial sequence of events is 1, 2, 3
Q2: partial sequence of events is 4, 5, 6
Q3: partial sequence of events is 2, 1
R1: partial sequence of events is 4, 3
R2: partial sequence of events is 4, 3, 2, 1
R3: partial sequence of events is 3, 2, 1
R4: partial sequence of events is 6, 5

The objective of the organization design problem is to decompose *RB*1 into five (possibly disjoint) sets of rules, where each set can be assigned to a decision maker (DM) in the organizational hierarchy presented in Fig. 2, which also shows the possible interactions among DMs. It is assumed that the physical architecture of the organization provides these DMs with the means to communicate with each other whenever required by the rules. At first, the decomposition of the rule base is done in a *vertical* manner (Mesarovic *et al.*, 1970); the decision rules are decomposed into three *layers* of subrules of increasing complexity. In the decomposed rule base *RB*2 rules of the form '$Ri \wedge Rj \wedge \ldots \rightarrow \_$' represent the rules assigned to DM1. The R*i* represent the responses of the lower-level decision makers communicated to DM1. Similarly, the set of rules defined at the intermediate layer is assigned to DM2, where the rules are of the form '$Qi \wedge Qj \wedge \ldots \rightarrow \_$'. Finally, the set of rules at the lowest layer of the rule base is further decomposed *horizontally* (partitioned) into three sets, and is assigned to three decision makes, DM3, DM4 and DM5, where the rules

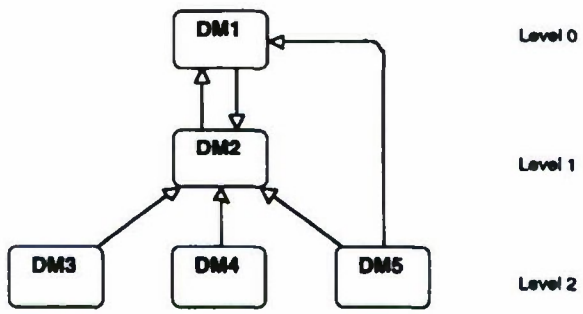


Fig. 2. Organizational hierarchy.

are of the form '$Pi \wedge Pj \wedge \ldots, \_$'. The decomposition of the original rule base is done by taking into account the fact that the set of basic concepts U can be divided into the following three subsets (not necessarily disjoint), where each set represents information from a different sector (area of awareness) assigned to a decision maker (DM3, DM5, and DM4 respectively):

$$U_1 = \{P1, P2, P3, P4, P5\}$$
$$U_2 = \{P6, P7, P8, P9, P10\} \quad (3)$$
$$U_3 = \{P8, P9, P10, P11, P12\}$$

The rules in *RB*2 correspond to the decomposition of the following rules in *RB*1: Rule2, Rule3 and Rule4 in *RB*2 correspond to the decomposition performed on Rule2 of *RB*1; Rule6, Rule7 and Rule8 in *RB*2 represent decomposition of Rule4 of *RB*1; Rule8, Rule9, Rule10 and Rule11 in *RB*2 correspond to Rule5 of *RB*1; Rule14, Rule15, Rule16 and Rule17 in *RB*2 correspond to Rule8 of *RB*1; Rule18, Rule19 and Rule20 in *RB*2 represent decomposition of Rule9 of *RB*1; and, finally, Rule21 and Rule22 of *RB*2 correspond to Rule10 of *RB*1. The rest of the rules are left as they were; Rule1, Rule5, Rule12, Rule13 and Rule23 in *RB*2 correspond to Rule1, Rule3, Rule6, Rule7 and Rule11 in *RB*1 respectively.

From now on, any reference to the rules implies the rules in RB_2 unless otherwise stated.

*Rule Base, RB*2.

Rule1: $P1 \wedge P2 \wedge P3 \rightarrow Q1$
Rule2: $P1 \wedge P2 \rightarrow \neg Q1$
Rule3: $P6 \wedge P7 \rightarrow Q4$
Rule4: $\neg Q1 \wedge Q4 \rightarrow R2$
Rule5: $P4 \wedge P5 \rightarrow Q3$
Rule6: $P3 \rightarrow Q10$
Rule7: $Q10 \wedge Q3 \rightarrow R3$
Rule8: $P4 \wedge P5 \rightarrow Q3$
Rule9: $P9 \wedge P12 \rightarrow Q5$
Rule10: $P9 \rightarrow Q6$
Rule11: $Q3 \wedge Q5 \rightarrow R3$
Rule12: $R2 \wedge R4 \rightarrow C$
Rule13: $P7 \wedge P9 \wedge P10 \rightarrow Q2$
Rule14: $P8 \wedge P9 \rightarrow Q5$
Rule15: $R1 \wedge R3 \rightarrow Q3$
Rule16: $Q3 \wedge Q5 \rightarrow R5$
Rule17: $R5 \rightarrow A$
Rule18: $P11 \wedge P12 \rightarrow Q2$
Rule19: $P8 \rightarrow Q8$
Rule20: $Q2 \wedge Q8 \rightarrow R4$
Rule21: $Q1 \wedge Q2 \rightarrow R6$
Rule22: $R6 \rightarrow B$
Rule23: $P6 \wedge P7 \rightarrow R1$

where:

Q4: partial sequence of events is 4, 3
Q5: partial sequence of events is 5, 6
Q6: P9 observed
Q8: P8 observed
Q10: P3 observed
R5: sequence of events is 4, 5, 6, 3, 2, 1
R6: sequence of events is 1, 2, 3, 4, 5, 6

For illustration purposes, the following two sets of mutually exclusive propositions are also defined:

$$\mu_1 = \{P8, P10\}$$
$$\mu_2 = \{R1, R5\} \quad (4)$$

The decomposition process by no means ensures the fact that the addition of new rules representing the replaced ones has not introduced errors. The effect of these new rules on the rest of the rule base can not be determined unless they are checked against the entire set of rules. This is where the methodology for the detection of problematic and erroneous cases is needed: one started with a correct rule base, decomposed it and assigned different rules to different decision makers in the organization, and as part of this process the resulting rule base no longer holds the property of being correct (at least it can not be claimed as correct). Since problematic cases may involve rules across rules assigned to individual DMs, the rule base *RB*2 is considered as a whole.

3. PROBLEMATIC CASES

A description of problematic and erroneous cases that need to be identified in a set of decision rules is presented in this section. Formal descriptions of the notions of incompleteness, inconsistency and redundancy are given.

3.1. *Redundant rules*

Redundancy in a rule base refers to the presence of multiple copies of the same rule or the presence of sets of rules that have the same effect (output) when initiated.

A trivial example of redundant rules is the following:

$$(p1 \wedge p2 \rightarrow A)$$
$$(p2 \wedge p1 \rightarrow A)$$

The two rules are identical except for the fact that their predicates in the antecedent are arranged differently, making it difficult to identify such cases by direct inspection or pattern matching. Another nontrivial example of redundant cases is as follows, where the last three rules together have the same effect as the first; such cases are very difficult to detect,

especially when these rules are spread out in a large rule base:

$$(q1 \wedge q2 \wedge q3 \wedge q4 \rightarrow A)$$

$$(q1 \wedge q2 \rightarrow p1)$$
$$(q3 \wedge q4 \rightarrow p2)$$
$$(p1 \wedge p2 \rightarrow A)$$

3.2. *Subsumed rules*

A rule base may contain two rules with identical conclusions where the antecedent conditions of one rule are a subset of the antecedent conditions of another. The first rule is said to subsume the second, since all the cases covered by the second rule are already covered by the first. The following is an example of such a case. Consider the two rules

$$(p1 \rightarrow A)$$
$$(p1 \wedge p2 \rightarrow A)$$

The first rule subsumes the second and makes it inactive. The definition of subsumed rules can be extended to two sets of rules where one set of rules subsumes the other.

3.3. *Inconsistent (conflicting) rules*

Definition Consistency (Zhang & Nquyen, 1989). A rule base is defined to be *consistent* if and only if there is no way of reaching contradictory assertions from *valid* input data.

A number of inconsistent cases have been reported in the literature (Nguyen *et al.*, 1987; Stachowitz and Combs, 1987; Zhang and Nguyen, 1989, 1992). However, in this paper, only the *conflicting rules* are considered as inconsistent. In some papers (Suwa *et al.*, 1982; Nguyen *et al.*, 1987; Zhang and Nguyen, 1989, 1992), subsumed and redundant rules are also considered as inconsistent. The following three cases of inconsistent rules are considered.

Case I: *Direct contradiction.* A set of rules is *inconsistent* if, by applying these rules, one could reach from a predicate p an assertion q, where p and q belong to a set of mutually exclusive predicates.

In this illustration a set of mutually exclusive predicates is denoted by μ. A generalized representation of such an inconsistency is

$$(\alpha \rightarrow \beta) \quad \text{where } \alpha, \beta \in \mu \qquad (5)$$

The set μ can be defined syntactically, i.e. $\beta = \neg\alpha$, or semantically by an expert. This definition of the set μ makes inconsistency a more stringent problem than the one in formal logic (Ginsberg, 1990).

Example.

$$(q1 \wedge q2 \rightarrow p1)$$
$$(q3 \wedge q4 \rightarrow p2)$$
$$(p1 \wedge p2 \rightarrow \neg q1)$$

Case II: *Contradiction in conclusion.* Rules are conflicting if their antecedents are the same, but the conclusions are mutually exclusive. A generalized representation of such a case is as follows:

$$(\alpha \rightarrow \beta)$$
$$(\alpha \rightarrow \tau) \text{ where } \beta, \tau \in \mu \qquad (6)$$

Example.

$$(p1 \wedge p2 \rightarrow p3)$$
$$(p3 \wedge p4 \rightarrow A)$$

$$(p1 \wedge p2 \wedge p4 \rightarrow \neg A)$$

Case III: *Contradiction in input.* Rules are conflicting if an assertion can be made through conflicting premises. The following expression presents the case:

$$(\alpha \wedge \beta \rightarrow \tau) \quad \text{where } \alpha, \beta \in \mu \qquad (7)$$

Example.

$$(p1 \wedge p2 \rightarrow p3)$$
$$(p3 \wedge \neg p1 \rightarrow A)$$

One final note is that the semantically defined mutually exclusive concepts cannot be identified by the system unless specified explicitly by the user.

3.4. *Circular rules*

This problem refers to the case of circular arguments. The following example illustrates the cases that are considered circular.

Example.

$$(p1 \rightarrow p2)$$
$$(p2 \rightarrow p3)$$
$$(p3 \rightarrow p1)$$

Circular rules may result in an infinite execution of rules unless an exit condition is also provided. Another case of circular rules is as follows:

$$(p1 \rightarrow p2 \wedge p3)$$
$$(p2 \wedge p4 \rightarrow p5)$$
$$(p3 \wedge p5 \rightarrow p4)$$

In the case where p1 is given as the input, the above set of rules results in a situation where the second rule cannot execute unless assertion p4 is obtained through the execution of the third rule, which in turn requires execution of the second rule for the input condition p5—a *deadlocked* situation.

3.5. *Incompleteness*

Definition: Incompleteness (Levesque, 1984). A knowledge base is *incomplete* when it does not have the information necessary to answer a question (*appropriately*) of interest to the system.

The following two types of incompleteness are reported in the literature (Zhang and Nguyen, 1992).

A. *Rules with ambiguous conditions* (*ambiguous rules*). A rule base is incomplete if, given a valid input, the system cannot interpret it in terms of applicable conditions in order to arrive at a conclusion. A cause of such incompleteness is the presence of at least one complex concept in the premise that cannot be explained or defined in terms of the basic concepts, U:

$$(A \wedge p1 \rightarrow R)$$

A is not the basic concept, and there exist no rules that explain the assertion A in terms of basic concepts, i.e. p2, p3 $\rightarrow$ A, where p2, p3 $\in$ U.

B. *Rules with useless conclusions* (*useless rules*). The main concept of a rule base can consist of a number of decision attributes that are derived by the rules. Given an input, if the conclusion derived is not the main concept or some or all of its attributes cannot be concluded from the rule base then the set of rules applicable to this input is incomplete. The main concept is domain-dependent and requires explicit definition:

$$(p1 \wedge p2 \rightarrow R)$$

R is not the main concept, and there is no rule going from R to an assertion α, where $\alpha \in \Psi$.

C. *Isolated rules*. A rule is an isolated rule if and only if

- All the propositions in its premise are complex concepts that cannot be explained in terms of the basic concepts, U; and
- its conclusion R is not the main concept, nor does there exist a rule (or rules) taking R to an assertion α, where $\alpha \in \Psi$.

4. PETRI NET REPRESENTATION OF RULES

An individual rule of the form 'P1 $\wedge$ P2 $\wedge \ldots \wedge$ P$n \rightarrow$ Q' is transformed to a PN with a single transition with n input places, each representing a single input proposition Pi, and an output place representing the assertion Q (for a general and more detailed description of this technique, see Zaidi, 1994). The labels of the places and transitions correspond to the propositions and rules they represent. The rule of inference is implemented by the execution mechanism of PN theory. A token in a place represents the truth assignment of a proposition. If all the places in the preset of a transition representing the premise of a rule being satisfied are marked then the rule (transition) is enabled and can execute (fire) making all the consequents (output places) valid (marked). The entire set of decision rules is then obtained by *unifying* all the individual rules. The process represents the causal relationship among the rules and the facts of the knowledge base. This method unifies the rules by merging all the places with identical labels (proposition symbols) into a single place. The PN obtained as a result of applying this technique to *RB*2 is shown in Fig. 3 where the basic inputs and main concepts defined for the rule base are shown aggregated into *virtual inputs and outputs*, P_{in} and P_{out}, with transitions T_{in} and T_{out} respectively. The parts of the net in the figure that are drawn by a broken line represent this aggregation, and do not represent any rule in the rule base.

It can be observed in Fig. 3 that the Petri net representation of a rule base does not provide an explicit relation between a predicate p and its negation $\neg$p. This lack of syntactic relation between predicates and their negations results in the absence of an implicit representation of the following axion (tautology):

$$((\alpha \rightarrow \beta) \rightarrow (\neg\beta \rightarrow \neg\alpha)) \qquad (8)$$

where α and β are well-formed formulae of Propositional Calculus.

An obvious solution to this problem is to include all the implications of the decision rules in *RB*2, before transforming it to an equivalent

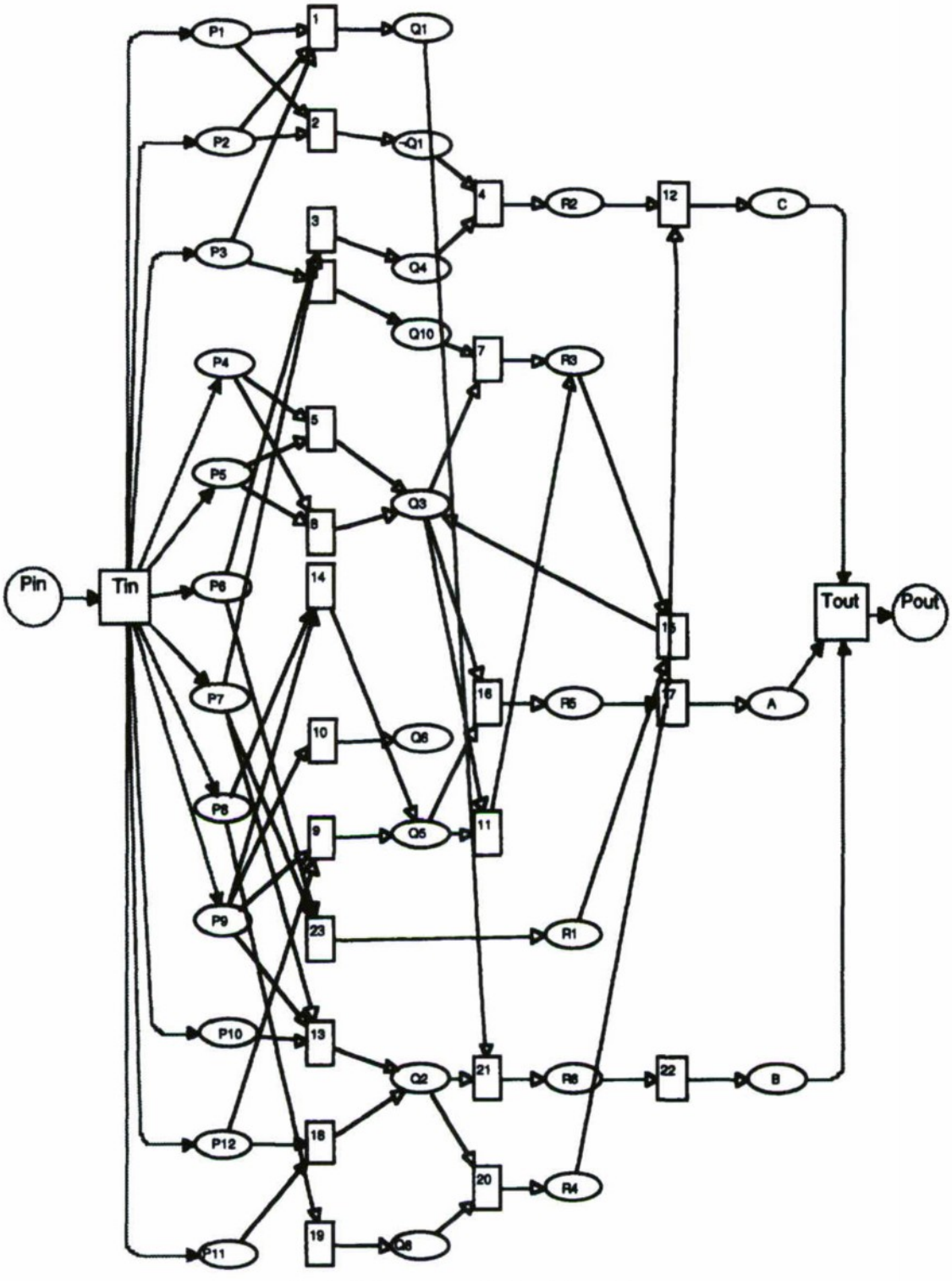


Fig. 3. Petri net representation of the rule base.

Petri net representation. The advantage of putting all the implications in the set of decision rules is that it will make the analyses independent of the form of the original rule base. The corresponding Petri net will be called an enhanced Petri net (EPN). However, such an approach will double the size of the rule base, and increase the size of the Petri net representation (not necessarily with the same ratio). Zaidi (1994) presented a graph-based approach that *normalizes* the EPN by removing certain *unnecessary* nodes present in the net. The parts of the EPN that help reveal the problematic cases are the only structures required, and the rest of the net can be considered unnecessary. The algorithm that normalizes the EPN requires the use of two algorithms, called *FPSO* and *FPSI*, which are

two variants of an earlier FindPath algorithm (Jin, 1986). The *FPSO*(p) algorithm, when applied to a node p in a PN, collects al the nodes that have directed paths to p and returns a subnet whose nodes are those collected by the algorithm. The *FPSI*(p) algorithm, on the other hand, collects all the nodes to which p has a directed path and returns a subnet composed of those nodes. Tables 2 and 3 present the formal descriptions of the two algorithms. The notation $x \twoheadrightarrow p$ used in the tables represents the binary relation that a directed path exists from node x to node p. The ordinary Petri net PN is defined by the 4-tuple (P, T, I, O), where P is the set of places and T the set of transitions, while I and O are the input and output relations that define the arcs.

The normalized EPN represents that part of the rule base that is influenced by the valid inputs and influences the output of the system. The rules in the rest of the net are considered unnecessary for the obvious reason. The net presented in Fig. 3 is a normalized net.

5. STATIC ANALYSIS

5.1. *Detection of incomplete rules*

The incompleteness, as defined in this paper, is determined by the lack of certain connections in the PN representation.

The isolated rules can be easily detected by a mere inspection of the normalized EPN; a rule in *RB*2 will be an isolated rule if the transition t, representing the rule, and the transition t', representing the implication obtained through the expression (5) are both absent in the net in Fig. 3. In the illustration, no isolated rules are found.

On the other hand, some of the obvious incomplete cases can be detected by simply searching the net for dangling places—places that have either only inputs (sources) or only outputs (sinks). The following algorithms provide a comprehensive approach to detecting and identifying incompleteness.

Table 2. FindPath-to-Sources (FPSO) algorithm

For a PN = (P, T, I, O)
$p \in V$ ($= P \cup T$)

$$FPSO(p) = (S, PN')$$

where

$$S = \{x \mid x \twoheadrightarrow p\}$$
$$PN' = (P', T', I', O')$$
$$PN \supseteq PN'$$
$$P \supseteq P'$$
$$T \supseteq T'$$
$$I \supseteq I'$$
$$O \supset O' \text{ and } P' \cup T' = S$$

Table 3. FindPath-to-Sinks (FPSI) algorithm

For a PN = (P, T, I, O)
$p \in V$ ($= P \cup T$)

$$FPSI(p) = (S, PN')$$

where

$$S = \{x \mid p \twoheadrightarrow x\}$$
$$PN' = (P', T', I', O')$$
$$PN \supseteq PN'$$
$$P \supseteq P'$$
$$T \supseteq T'$$
$$I \supseteq I'$$
$$O \supseteq O' \text{ and } P' \cup T' = S$$

Algorithm for ambiguous rules.

- Apply *FPSI* to P_{in}, and compare the output of *FPSI*(P_{in}) with the original net. Those places of the original net that do not appear in the output of *FPSI*(P_{in}) identify ambiguous rules.

Algorithm for useless rules.

- Apply *FPSO* to P_{out}, and compare the output of *FPSO*(P_{out}) with the original net. Those places of the original net that do not appear in the output of *FPSO*(P_{out}) identify useless rules.

The application of these algorithms resulted in Rule10 being identified as a useless rule, since Q6 (P9 observed) appeared as a dangling place with only input arcs (a sink). The transition in the preset of this assertion identifies the corresponding useless rule; the rule's consequent represent the assertion that the system might infer (from the basic inputs to the system) but cannot interpret in terms of the required outputs.

5.2. *S-invariant analysis*

The S-invariant analysis looks at all the directed paths in the PN and searches them for problematic cases. The analysis is shown to reveal certain patterns of PNs that correspond to circular and inconsistent rules. Before a detailed description of the analysis is presented, the following definitions are in order.

Definition: *S-invariant.* Given an incidence matrix C of a PN, an *S-invariant* is a $n \times 1$ non-negative integer vector X of the kernel of C^T, i.e.

$$C^T X = 0 \tag{9}$$

Definition: *Support of S-invariant.* If X is an

S-invariant, the set of places whose corresponding components in X are strictly positive is the *support* of the invariant, denoted by $\langle X \rangle$.

The support of an S-invariant is said to be *minimal* if and only if it does not contain the support of another S-invariant but itself and the empty set.

Definition: *S-component.* The *S-component* associated with an S-invariant X of a Petri Net is the subnet whose places are the places of $\langle X \rangle$ and whose transitions are the input and output transitions of the places of $\langle X \rangle$.

By extension, a *minimal S-component* is the S-component of a minimal support S-invariant.

Definition: *marked graph.* A *marked graph* is a connected Petri net in which each place has exactly one input and one output transition.

The minimal S-invariants of the PN are calculated after it is transformed to an equivalent marked graph representation. The algorithm to convert a PN into a marked graph Petri net (MPN) is given as follows.

- Merge the virtual input and output places, P_{in} and P_{out}, into a single external place P_e.
- $\forall p$ so that $|{}^{\cdot}p| = m$ (>1) and $|p^{\cdot}| = n(>1)$, create $m \times n$ copies of p, denoted by p^i, where $i = 1, 2, \ldots, m \times n$. Create n *links* from each of m transitions, $ti \in {}^{\cdot}p$, to all of the n transitions, $tj \in p^{\cdot}$, through these copies of the place p. The process will create $m \times n$ links between the input and output transitions of place p. (Figure 4 illustrates the process.)

A place p is said to constitute a link from a transition ti to another transition tj if ${}^{\cdot}p = \{ti\}$ and $p^{\cdot} = \{tj\}$. The process of converting a PN to a MPN preserves the connectivity of the original PN in the sense that if two nodes are directly connected in PN then they will remain connected in the MPN. The net in Fig. 4 is converted to an equivalent MPN representation, and the S-invariants are calculated for it. (Table 4 lists the supports of the calculated minimal S-invariants). The following theorem characterizes the S-components corresponding to the minimal support S-invariants of the MPN.

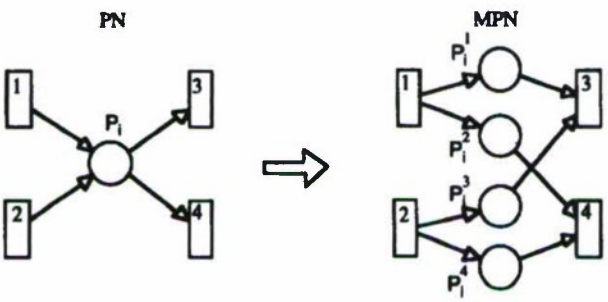


Fig. 4. Process of converting a PN to MPN.

Table 4. Minimal supports corresponding to S-invariants of the MPN

j	Support of S-invariant $\langle Xi \rangle$
1	{$Q3^1$, $R3^1$}
2	{$Q3^2$, $R3^2$}
3	{Pe, $P1^1$, Q1, R6, B}
4	{Pe, $P1^2$, ¬Q1, R2, C}
5	{Pe, $P2^1$, Q1, R6, B}
6	{Pe, $P2^2$, ¬Q1, R2, C}
7	{Pe, $P3^1$, Q1, R6, B}
8	{Pe, $P3^2$, Q10, $R3^1$, $Q3^3$, R5, A}
9	{Pe, $P4^1$, $Q3^4$, $R3^1$, $Q3^3$, R5, A}
10	{Pe, $P4^1$, $Q3^5$, R5, A}
11	{Pe, $P4^1$, $Q3^6$, $R3^2$, $Q3^2$, R5, A}
12	{Pe, $P4^2$, $Q3^7$, $R3^1$, $Q3^3$, R5, A}
13	{Pe, $P4^2$, $Q3^8$, R5, A}
14	{Pe, $P4^2$, $Q3^9$, $R3^2$, $Q3^3$, R5, A}
15	{Pe, $P5^1$, $Q3^4$, $R3^1$, $Q3^3$, R5, A}
16	{Pe, $P5^1$, $Q3^5$, R5, A}
17	{Pe, $P5^1$, $Q3^6$, $R3^2$, $Q3^3$, R5, A}
18	{Pe, $P5^2$, $Q3^7$, $R3^1$, $Q3^3$, R5, A}
19	{Pe, $P5^2$, $Q3^8$, R5, A}
20	{Pe, $P5^2$, $Q3^9$, $R3^2$, $Q3^3$, R5, A}
21	{Pe, $P6^1$, Q4, R2, C}
22	{Pe, $P6^2$, R1, $Q3^3$, R5, A}
23	{Pe, $P7^1$, Q4, R2, C}
24	{Pe, $P7^2$, R1, $Q3^3$, R5, A5}
25	{Pe, $P7^3$, $Q2^1$, R6, B}
26	{Pe, $P7^3$, $Q2^2$, R4, C}
27	{Pe, $P8^1$, $Q5^1$, R5, A}
28	{Pe, $P8^1$, $Q5^2$, $R3^2$, $Q3^3$, R5, A}
29	{Pe, $P8^2$, Q8, R4, C}
30	{Pe, $P9^1$, $Q5^1$, R5, A}
31	{Pe, $P9^1$, $Q5^2$, $R3^2$, $Q3^3$, R5, A}
32	{Pe, $P9^2$, $Q5^3$, R5, A}
33	{Pe, $P9^2$, $Q5^4$, $R3^2$, $Q3^3$, R5, A}
34	{Pe, $P9^3$, $Q2^1$, R6, B}
35	{Pe, $P9^3$, $Q2^2$, RA, C}
36	{Pe, P10, $Q2^1$, R6, B}
37	{Pe, P10, $Q2^2$, R4, C}
38	{Pe, $P12^1$, $Q5^3$, R5, A}
39	{Pe, $P12^1$, $Q5^4$, $R3^2$, $Q3^3$, R5, A}
40	{Pe, $P12^2$, $Q2^1$, R6, B}
41	{Pe, $P12^2$, $Q2^2$, R4, C}
42	{Pe, P11, $Q2^1$, R6, B}
43	{Pe, P11, $Q2^2$, R4, C}

Definition: *Directed circuit.* A *directed circuit* is a directed path from one node back to itself.

A *directed elementary circuit* is a directed circuit in which only one node appears more than once.

Theorem 1. (Hillion (1986).) The minimal S-components of a marked graph Petri net are exactly its directed elementary circuits.

Based on the results from Theorem 1, the following proposition characterizes two different types of circuits in the MPN.

Proposition 1. Let $\langle Xi \rangle$ be the minimal support of the calculated S-invariant Xi.

- If $\langle Xi \rangle$ contains the external place P_e then the S-component associated with $\langle Xi \rangle$ represents a directed path from a basic input to one of the main concepts.
- If $\langle Xi \rangle$ does not contain the external place Pe then the S-component associated with $\langle Xi \rangle$ represents a loop inside the Petri net structure.

The following proposition interprets the minimal supports calculated for the MPN in terms of circular and possible inconsistent rules. The notation $p^{(\cdot)}$ used in the proposition indicates the fact that the place p might be replicated during the process of converting PN to MPN. The notation illustrates that the algorithm only identifies the predicate p with no regard to the superscript assigned to it during the conversion process.

Proposition 2.

- If there exists a $\langle Xi \rangle$ that contains place $p^{(\cdot)}$ and $q^{(\cdot)}$, where p and $q \in \mu$ (set of mutually exclusive concepts) then $\langle Xi \rangle$ indicates the presence of inconsistent rules in the rule base. The S-components associated with $\langle Xi \rangle$ identify the set of inconsistent rules.
- If there exist $\langle Xi \rangle$ and $\langle Xj \rangle$, where $p^{(i)} \in \langle Xi \rangle$ and $q^{(j)} \in \langle Xj \rangle$, p and $q \in \mu$, and $\langle Xi \rangle \cap \langle Xj \rangle \neq \varnothing$, then $\langle Xi \rangle$ and $\langle Xj \rangle$ indicate the presence of *possible* inconsistent rules in the rule base. The S-components associated with $\langle Xi \rangle$ and $\langle Xj \rangle$ identify the set of inconsistent rules.
- If there exists a $\langle Xi \rangle$ that does not contain the external place Pe then $\langle Xi \rangle$ indicates the presence of circular rules in the rule base. The transitions in the S-component associated with $\langle Xi \rangle$ identify the set of circular rules.

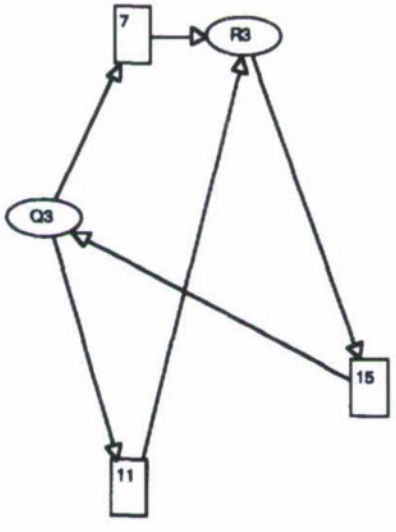


Fig. 5. Circular cases.

The first case presented in Proposition 2 corresponds to rules where it is possible to reach from a predicate 'p' to its negation or to a mutually exclusive concept 'q', which is not permissible in formal logic. The second inconsistent case presented corresponds to a rule that requires conflicting assertions to be simultaneously true in order to excute.

The application of this analysis performed on the PN representing *RB*2 and *RB*2 resulted in the following problematic cases.

Circular rules. $\langle X1 \rangle$ and $\langle X2 \rangle$ indicate the presence of the following circular rules:

Rule15 and Rule7;
Rule15 and Rule11

Figure 5 shows the reported circular cases in the PN. In one of the reported circular cases (Rule15 and Rule7), the system infers R3 (partial sequence of events in 3, 2, 1) through Q10 (P3 observed) and Q3 (partial sequence of Events is 2, 1) as illustrated by Rule7. On the other hand, Rule15 requires Q3 to infer R3. Similarly, in the second case (Rule15 and Rule11), Rule11 infers R3 through Q3, where Q3 is required to be inferred through R3 in Rule15.

The analysis of *RB*2 reveals that the reported circular cases were introduced during the decomposition of Rule4, Rule5 and Rule8 of the original rule base *RB*1.

• *Inconsistent rules.* Supports $\langle X22 \rangle$ and $\langle X24 \rangle$ indicate the presence of an inconsistent case, since both have R1 (partial sequence of events is 4, 3) and R5 (sequence of events is 4, 5, 6, 3, 2, 1) present in them, where R1 and R5 are defined as mutually exclusive concepts. The following rules correspond to this inconsistency: Rule15 and Rule16.

The inconsistent rules, Rule15 and Rule16 ($R1 \wedge R3 \rightarrow Q3$ and $Q3 \wedge Q5 \rightarrow R5$), were introduced in the rule base as a result of the decomposition of Rule8 of the original rule base *RB*1. Supports $\langle X29 \rangle$ and $\langle X37 \rangle$ indicate the presence of another inconsistent case. The S-components corresponding to these supports reveal that Rule20 requires both P10 and P8 to be present simultaneously in order to fire. P10 and P8 are defined as mutually exclusive concepts, and can be true at the same time. Therefore the following inconsistent rules are found; Rule13, Rule19 and Rule20.

Rule19 and Rule20 ($P8 \rightarrow Q8$ and $Q2 \wedge Q8 \rightarrow R4$) were added to the rule base as a result of the decomposition of Rule9 of the original rule base, while Rule13 already existed in the rule base.

6. DYNAMIC ANALYSIS

The dynamic analysis of the PN is performed to explore the behavioral properties of the graph. The specific technique presented in this section is the occurrence graph (OG) analysis. The analysis considers all the possible states of a net that can be reached from an initial state (marking) after a finite number of firings of transitions in the net.

Definition: *Marking* (*state*). A *marking* (*state*) of a Petri net is a mapping $M: P \rightarrow \{0, 1, 2, \ldots\}$ that assigns a non-negative integer (the number of tokens) to each place.

Definition: *Occurrence graph.* An *occurrence graph* associated with a Petri net and an initial marking M^0 represents all possible reachable markings from M^0. Each node of the occurrence graph represents a reachable marking, and the arcs between nodes represent transitions from one marking to another (with arc annotations indicating the transition name).

For a given input, the analysis can generate an occurrence graph for the PN, and the occurrence graph can be searched for states that can be reached from more than one distinct paths in the OG—*redundant cases*—and for conflicting states reachable from a single input—*conflicting cases*. But, with no knowledge of the feasible input domain, one is left with 2^n possible combinations of input vectors ($n = |U|$). The construction of these many OGs is simply too large a problem. An approach is presented to overcome this problem.

The redundancies present in a rule base appear in the PN representation as places with multiple inputs representing the redundancy in arriving at a conclusion, and with places with multiple outputs representing several paths originating from the same input. This heuristic is used to determine whether or not the PN representing *RB*2 should be analyzed by the OG analysis for the redundant cases.

The heuristic can also be used to narrow down the search for redundancies to only those parts of the PN with such places. The following steps, performed on the PN, extracted the problematic parts of the original PN.

- For the PN construct a set A;

  $A = \{p \mid p \in$ set of places in PN and $|{}^{\cdot}p| > 1\}$.

- From A select a place p_i (starting from the elements of the set of main concepts Ψ), and apply $FPSO(p_i)$. The result is denoted by PN_i.
- If PN_i contains places with multiple output arcs, put PN_i in a set RN. Otherwise, declare PN_i free of redundant cases.
- Remove from A all the places that are present in PN_i. Repeat the process as long as there are elements in A.

The technique is applied to the PN representing *RB*2. Figure 6 shows the elements of RN.

The only thing remaining is the determination of initial markings before any OGs can be constructed and searched for states that correspond to the problematic cases. The solution to this problem reverses the direction of all the arcs in each PN_i. The idea is to scan the feasible valid input space from a known and much smaller set of outputs. The resulting net is denoted by PN_i' and defined as $PN_i' = (P, T, O, I)$, where PN_i is given by the tuple (P, T, I, O). The modified net is then initialized by putting a single token in the place p_i (note that p_i is the place on which the FPSO algorithm was applied that resulted in the determination of PN_i). Since all the nets in RN are Y-nets, every transition in the PN_i' will have only one input; a single token in place p_i is enough to ensure that every place in the PN_i' will appear as marked at least once in the occurrence graph constructed for PN_i'. For PNs that are not Y-nets, an algorithm (Zaidi, 1994) is used to convert them to an equivalent Y-net representation. It has been shown that the OGs constructed for the modified nets PN′ exhibit the behavioral properties that can be interpreted in terms of the properties of the original net PN (Zaidi, 1994). The following propositions now states the results of the OG analysis applied to the nets in RN.

Proposition 3: *Redundant rules.* Let $PN_i \in$ RN. Distinct firing vectors corresponding to multiple paths between two nodes of OG_i, or paths from one root node to two or more nodes having the same set (or a subset) of places as marked, identify redundant rules. The transitions whose corresponding components in the firing vectors are positive represent the rules that are abundant.

The algorithm proceeds by reversing the arrows of the nets PN_i in set RN and initializing the modified nets, termed PN_i', by putting a token in source places. The occurrence graphs OG_i are constructed for each PN_i. The occurrence graphs so obtained are searched for redundancies. The corresponding occurrence graphs are shown in Fig. 7. The following problematic cases are found.

Redundant rules. Rule5 and Rule8 are redun-

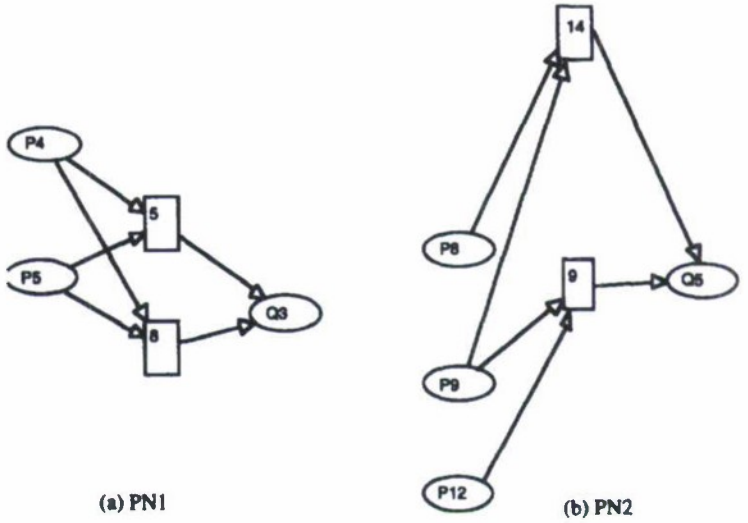


Fig. 6. Elements of the set RN.

dant rules, since they represent two copies of a single rule present simultaneously in *RB*2.

An investigation of the original rule base and the decomposition process reveals that the rule $P4 \wedge P5 \rightarrow Q3$ already existed in the original rule base, and another copy of this rule was added to the set of decision rules as a result of the decomposition of Rule5 of *RB*1. It turns out that an inconsistent case, in which conflicting assertions can be made through a single input, can be detected by a methodology similar to the one used for the redundancies. The procedure is described as follows.

- For a set of mutually exclusive concepts μ_i present in the rule base, merge all the places in μ into a single virtual place p_i.
- Apply $FPSO(p_i)$. The result is denoted by PN_i.
- If PN_i contains places with multiple output arcs, put PN_i in a set CN. Otherwise, declare PN_i free of conflicting rules.
- Repeat for all sets of mutually exclusive concepts defined for the system in hand, except for those sets whose elements have already been identified in the conflicting rules reported by the static analyses.

The technique extracts the subnets of the PN with possible inconsistencies. Figure 8 presents the resulting net. The subnet is then used as an input to the approach applied for detecting redundancy. The following proposition characterizes the conflicting cases that can be detected by the analysis.

Proposition 4: *Conflicting rules.* Let $PN_i \in CN$ and let there not exist any redundant case inside the decision rules represented by PN_i. Then application of the techniques used for redundancy and results from Proposition 3 on Occurrence graph OG_i identifies conflicting rules.

The OG in Fig. 9 identifies the following inconsistent case in *RB*2.

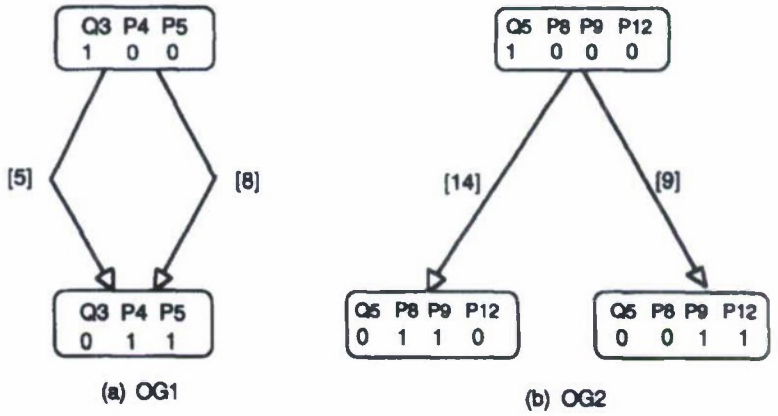


Fig. 7. Occurrence graphs.

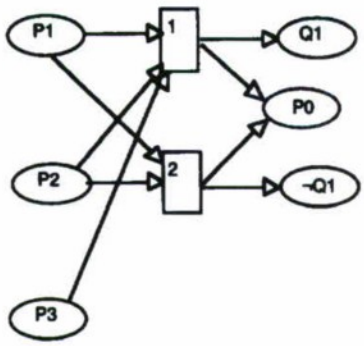


Fig. 8. Element of the set CN.

• *Inconsistent rules.* Rule1 and Rule2 are inconsistent, since OG_1 in Fig. 9 represents a redundant case. Rule1 infers Q1 by observing inputs P1, P2 and P3. However, Rule2 falsifies Q1 by observing a subset of the information that inferred the truth of Q1, thus creating an inconsistency.

7. CONCLUSION

The results of the example successfully illustrated the types of errors that might find their way into a set of decision rules during the decomposition of the rule base for assignment to several decision makers. The types of inconsistencies introduced during the decomposition of rules are common, especially when different rules are decomposed by different individuals: different people tend to interpret identidal inputs differently (sometimes oppositely) based on their assumptions about the problem and experience biases. The maintenance of consistency over a longer period of time becomes more and more difficult, especially when the size of the rule base is also growing rapidly. The illustrative example started with a smaller rule base, and resulted in a much larger set of decision rules. Some of the problematic cases were intentionally introduced to illustrate the capabilities of the methodology, but the resulting reported cases included cases that were not apparent during problem formulation.

The methodology for uncovering problematic rules in a rule-based system presented stands out among all the reported V&V methods because of its generality in terms of the constraints imposed on the representation of the rule base and its flexibility, since it can be applied at any stage of knowledge acquisition. A unique feature of the approach is that the inconsistencies are defined both *syntactically* and *semantically* (Ginsberg, 1988); however, semantic inconsistencies are defined with respect to a user-provided set of mutually exclusive concepts.

The approach presented in this paper can easily be extended to rule bases in first-order predicate calculus (Zaidi, 1994). The approach uses the colored Petri net formalism to solve the problem.

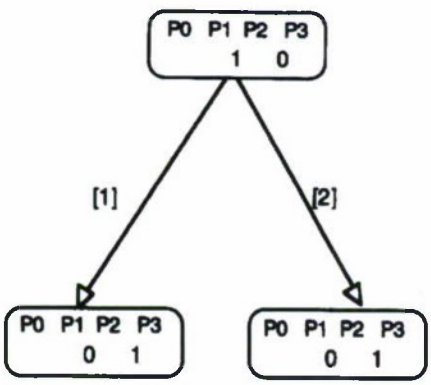


Fig. 9. Occurrence graphs for the element of CN.

REFERENCES

Agarwal, R. and M. Tanniru (1992). A Petri net based approach for verifying the integrity of production systems. *Int. J. Man. Machine Stud.*, **36**, 447–468.

Ayel, M. and J. Laurent (Eds) (1991). *Validation, Verification and Test of Knowledge-based Systems.* Wiley, Chichester.

Boehm, B. W. (1984). Verifying and validating software requirements and design enhancement. *IEEE Software*, January.

Charles, E. and O. Dubois (1991). MELODIA: logical methods for checking knowledge bases. In M. Ayel and J. Laurent (Eds), *Validation, Verification and Test of Knowledge-based Systems.* Wiley, Chichester.

Cragun, B. J. and H. J. Steudel (1987). A decision-table-based processor for checking completeness and consistency in rule-based expert systems. *Int. J. Man–Machine Studies*, **26**, 633–648.

Davis, R. (1976). Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases. PhD Dissertation, Department of Computer Science, Stanford University.

De Raedt, L., G. Sablon and M. Bruyoogh (1991). Using interactive concept learning for knowledge base validation and verification. In M. Ayel and J. Laurend (Eds), *Validation, Verification and Test of Knowledge-based Systems.* Wiley, Chichester.

Ginsberg, A. (1988). Knowledge-base reduction: a new approach to checking knowledge bases for inconsistency and redundancy. In *Proc. 7th AAAI Conf.*

Giordana, A. and L. Saitta (1985). Modeling production rules by means of predicate transition networks. *Inf. Sci.*, (35). 1–41.

Gupta, U. G. (Ed.) (1991). *Validating and Verifying Knowledge-based Systems.* IEEE Computer Society Press, Los Alamitos, CA.

Hillion, H. P. (1986). Performance evaluation of decision-making organizations using timed Petri nets. LIDS-TH-1590, Laboratory of Information and Decision Systems, MIT.

Jin, V. Y. (1986). Delays for distributed decisionmaking organizations. LIDS-TH-1459, Laboratory for Information and Decision Systems, MIT.

Lee, S. and R. M. O'Keefe (1994). Developing a strategy for expert system verification and validation. *IEEE Trans. Syst., Man., Cybernetics*, **SMC-24**, 643–655.

Lehner, P. E. (1989). Towards an empirical approach to evaluating the knowledge base of an expert system. *IEEE Trans. Syst., Man, Cybernetics*, **SMC-19**, 658–662.

Levesque, H. J. (1984). The logic of incomplete knowledge bases. In M. L. Brodie, J. Myopoulos and J. W. Schmidt (Eds), *On Conceptual Modeling: Perspective from Artificial Intelligence, Databases and Programming Languages*, pp. 165–189. Springer-Verlag, New York.

Levis, A. H., D. M. Perdu and A. K. Zaidi (1995). CAESAR II: computer aided evaluation of system architectures. Report GMU/C31-162-R, C31 Center, George Mason University, Fairfax, VA.

Lim, E. L., W. W. Tsang and J. McCallum (1986). A graph model for checking consistency and completeness of production rules. Technical Report, DISCS Publication TRB8/86, National University of Singapore.

Liu, N. K. and T. Dillon (1991). An approach towards the verification of expert systems using numerical Petri nets. *Int. J. Intelligent Syst.*, **6**, 255–276.

van Melle, W. (1980). A domain-independent system that aids in constructing knowledge-based consultation program. PhD Dissertation, Department of Computer Science, Stanford University.

Mesarovic, M. D., D. Macko and Y. Takahara (1970). *Theory of Hierarchical, Multilevel Systems*. Academic Press, New York.

Nguyen, T. A., W. A. Perkins, T. J. Laffey and D. Pecora (1987). Knowledge base verification. *AI Magazine*, **8**(2), 69–75.

O'Keefe, R., O. Balci and E. Smith (1987). Validating system performance. *IEEE Expert*, 81–89.

Smith, S. and A. Kandel (1993). *Verification and validation of Rule-Based Expert Systems*. CRC Press, Boca Raton, FL.

Stachowitz, R. A. and J. B. Combs (1987). Validation of expert systems. In *Proc. 20th Hawaii International Conf. on Systems Science (HICSS)*, Vol. 1, pp. 689–695.

Stickel, M. E. (1988). A Prolog technology theorem prover: implementation by an extended Prolog compiler. *J. Autom. Reasoning*, **4**, 353–380.

Suwa, M., A. C. Scott and E. H. Shortliffe (1982). An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine*, **3**(4), 16–21.

Suwa, M., A. C. Scott and E. H. Shortliffe (1985). Completeness and consistency in a rule-based system. In B. G. Buchanan and E. H. Shortliffe (Eds), *Rule-based Expert Systems*. Addison-Wesley, Reading, MA.

Wesson, R. and F. Hayes-Roth (1980). Network structures for distributed situation assessment. R-2560-ARPA, Rand Report, Santa Monica, CA.

Wilkins, D. C. and B. G. Buchanan (1986). On debugging rule sets when reasoning under uncertainty. *Proc. AAAI*, **1**, 448–454.

Yu, V. L., B. G. Buchanan, E. H. Shortliffe, S. M. Wraith, R. Davis, A. C. Scott and S. N. Cohen (1979). Evaluating the performance of a computer based consultant. *Comput. Prog. Biomed.*, **9**, 95–102.

Zaidi, S. A. K. (1994). Validation and verification of decision making rules. Technical Report GMU/C31-155-TH, C31 Center, George Mason University, Fairfax, VA.

Zhang, Du and D. Nguyen (1989). A technique for knowledge base verification. In *Proc. IEEE International Workship on Tools for AI (TAI'89)*, Fairfax, VA, pp. 399–406.

Zhang, Du and Nguyen, D. (1992). A tool for knowledge base verification. In *Development of Knowledge-based Shells*. World Scientific, Singapore.

Zisman, M. D. (1978). Use of production system for modeling asynchronous, concurrent process. In D. A. Waterman (Ed.), *Pattern Directed Inference Systems*. Academic Press, New York.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06-05-2011	Final Technical Report	01-16-2008 to 04-30-2011

4. TITLE AND SUBTITLE
Scalable Adaptive Architectures for Maritime Operations Center Command and Control

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-08-1-0319

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Wagenhals, Lee W.
Zaidi, Abbas K.
Levis, Alexander H.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
System Architectures Laboratory
Dept. of Electrical and Computer Engineering
George Mason University, Fairfax, VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER
SAL/FR-11-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
OFFICE OF NAVAL RESEARCH
875 N. RANDOLPH ST.
ONE LIBERTY CENTER
ARLINGTON VA 22203-1995

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Unclassified Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The objective was to analyze the Navy Maritime Operations Center (MOC) focusing on the scalability aspects and develop tools and techniques for the design and evaluation of augmentation strategies that will allow MOCs to adapt to new missions and new levels of command and control. A suite of tools was developed to support MOC scalability. Two basic modes of augmentation were defined and analyzed: augmentation by staff and augmentation by function. The second objective was to address the dynamically changing rules of engagement of a maritime force as it crosses different geographical areas. A formal model checking methodology was used to detect redundant and inconsistent cases in a set of rules selected from across different Maritime Laws. A software suite of tools was designed to provide Staff Judge Advocate with a repository of OPLAW/ROE sets of rules that enable information foraging and compiling capability. The selected sets of rules are then analyzed by the RULER tool for consistency and redundancy. A reasoning engine provides an "If-then' analysis capability for the final selection of applicable rules in given mission situation.

15. SUBJECT TERMS
Organization Design, Scalable Organizations, Colored Petri Nets, Rules of Engagement, Rule Validation, Maritime Law

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	SAR	81	Alexander H. Levis
Unclassified	Unclassified	Unclassified			19b. TELEPONE NUMBER *(Include area code)* 703 993 1619